# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-155845

(43)Date of publication of application : 06.06.2000

(51)Int.Cl. G06T 11/00
G06F 3/153
G06F 12/00
G06F 12/04
G06T 1/00

(21)Application number : 11-140238

(22)Date of filing : 20.05.1999

(71)Applicant : MITSUBISHI ELECTRONICS AMERICA INC

(72)Inventor : ELIZABETH J SHURAPPU

(30)Priority

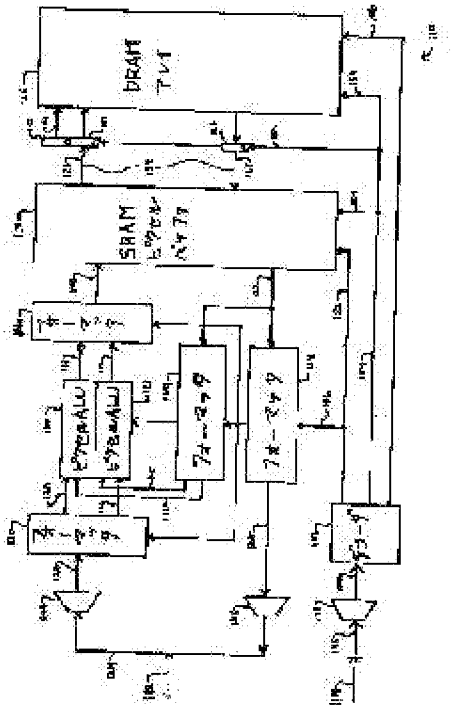
| Priority number : | Priority date : | Priority country : |
|---|---|---|
| 98 86554 | 21.05.1998 | US |
| 98 164858 | 01.10.1998 | US |
| 98 164866 | 01.10.1998 | US |
| 99 264261 | 08.03.1999 | US |
| 99 264281 | 08.03.1999 | US |

## (54) STORAGE DEVICE, DATA FORMATTER, METHOD FOR ACCESSING DATA, METHOD FOR CLEARING AREA OF DATA, METHOD FOR COMPRESSING DATA, METHOD FOR FORMATTING DATA, AND GRAPHIC SYSTEM OPERATION METHOD

(57)Abstract:

PROBLEM TO BE SOLVED: To obtain rendering which is made fast by composing a storage device using a plurality of system of pixel arithmetic-logic units(ALUs) which are coupled with a pixel buffer, an input/output data formatter, and a read/write data formatter.

SOLUTION: Input graphics data are demultiplexed by an input data demultiplexer 126, and transferred to the input data formatter 130 through an input data bus 128 and formatted, and the data are transmitted to pixel ALUs 120 and 121 for processing through input data buses 129 and 131. The data are formatted by a formatter 140 from a pixel ALU to an SRAM and allocated to an SRAM pixel buffer 118 through a data bus 141. The SRAM pixel buffer 118 reads out the data formatted by the formatter 140 from the pixel ALU to the SRAM through a pixel ALU data bus 138.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Your Ref: 07844-499JP1
Our Ref: PA1101

**Translation of Selected Portions of Pat. Laid-open Official Gazette**

---

Appln. No: 11-140238
Appln. Date: May 20, 1999
Laid-open Pub. No: 2000-155845
Laid-open Pub. Date: June 6, 2000
Priorities: 5/21/98 U.S.S.N. 60/086554, 10/1/98 U.S.S.N. 09/164858, 10/1/98 U.S.S.N. 09/164866, 3/8/99 U.S.S.N. 09/264261 & 3/8/99 U.S.S.N. 09/264281

Inventor(s): Elizabeth J Shulapp (?)
Applicant(s): Mitsubishi Electronics America Inc.
Attorney(s): Kuro Fukami et al.

---

1. Title of the Invention

MEMORY DEVICE, DATA FORMATTER, A METHOD FOR ACCESSING DATA, A METHOD FOR CLEARING A DATA REGION, A METHOD FOR COMPRESSING DATA, A METHOD FOR FORMATTING DATA, A GRAPHIC SYSTEM AND A METHOD FOR OPERATING A GRAPHIC SYSTEM

2. Claims

(omitted)

3. Detailed Description of the Invention (Selected Portions)

1)

(omitted)

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－155845
（Ｐ2000－155845Ａ）
(43)公開日　平成12年６月６日(2000.6.6)

| (51)Int.Cl.[7] | 識別記号 | ＦＩ | | テーマコード（参考） |
|---|---|---|---|---|
| Ｇ０６Ｔ　11/00 | | Ｇ０６Ｆ　15/72 | Ａ | |
| Ｇ０６Ｆ　3/153 | ３３６ | 3/153 | ３３６Ｂ | |
| 12/00 | ５８０ | 12/00 | ５８０ | |
| 12/04 | ５３０ | 12/04 | ５３０ | |
| Ｇ０６Ｔ　1/00 | | 15/66 | Ｊ | |

審査請求　未請求　請求項の数53　ＯＬ　（全 77 頁）

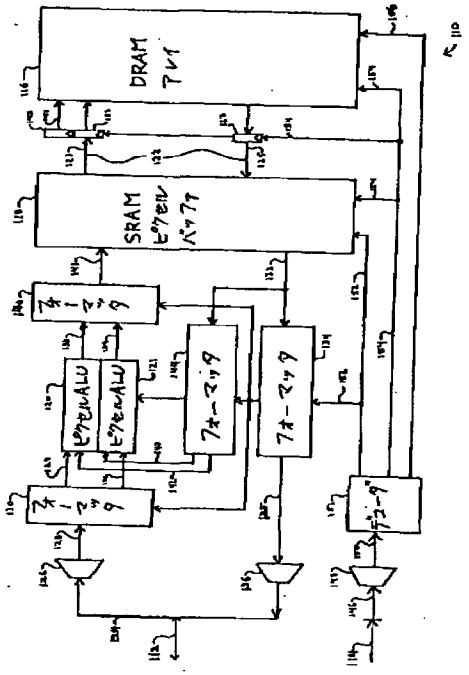
(21)出願番号　特願平11－140238
(22)出願日　平成11年５月20日(1999.5.20)
(31)優先権主張番号　６０／０８６５５４
(32)優先日　平成10年５月21日(1998.5.21)
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　０９／１６４８５８
(32)優先日　平成10年10月１日(1998.10.1)
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　０９／１６４８６６
(32)優先日　平成10年10月１日(1998.10.1)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　591260649
ミツビシ・エレクトロニクス・アメリカ・インコーポレーテッド
アメリカ合衆国、90630－0007　カリフォルニア州、サイプレス、プラザ・ドライブ、5665
(72)発明者　エリザベス・ジェイ・シュラップ
アメリカ合衆国、95132　カリフォルニア州、サン・ノゼ、ベクスレー・ランディング、1841
(74)代理人　100064746
弁理士　深見　久郎　（外３名）

最終頁に続く

(54)【発明の名称】　記憶装置、データフォーマッタ、データにアクセスする方法、データの領域をクリアする方法、データを圧縮する方法、データをフォーマット化する方法、グラフィックスシステムおよびグラ

(57)【要約】
【課題】　コンピュータグラフィックスシステムにおいて二次元および三次元イメージの増速されたレンダリングをもたらす半導体記憶装置を提供する。
【解決手段】　半導体記憶装置は、メモリアレイ（１１６）と、ピクセルバッファ（１１８）と、ピクセルバッファに結合される複数のピクセル算術論理演算装置（１２０、１２１）と、入力データフォーマッタ（１３０）と、出力データフォーマッタ（１３４）と、読出データフォーマッタ、書込データフォーマッタと、アドレスおよび制御入力バス（１４６）とを含む。



1

【特許請求の範囲】

【請求項１】　メモリアレイと、
前記メモリアレイに結合されるバッファと、
前記バッファに結合される複数の算術論理演算装置とを含む記憶装置であって、さらに
前記複数の算術論理演算装置に結合され、かつ前記記憶装置の外部にあるレンダリングバスに結合される入力データフォーマッタと、
前記バッファおよび外部レンダリングバスに結合される出力データフォーマッタと、
前記複数の算術論理演算装置および前記バッファに結合される書込データフォーマッタと、
前記バッファおよび前記複数の算術論理演算装置に結合される読出データフォーマッタと、
前記メモリアレイ、前記バッファ、前記複数の算術論理演算装置、前記入力データフォーマッタ、前記出力データフォーマッタ、前記書込データフォーマッタ
、前記読出データフォーマッタおよび、前記記憶装置の外部にあるアドレスおよび制御バスに結合されるアドレスおよび制御入力バスとを含む記憶装置。

【請求項２】　コントローラに結合されるメモリのためのデータフォーマッタであって、
前記コントローラから前記データフォーマッタへ送信される１つ以上の信号から１組のピクセル差成分を抽出する第１のフォーマッタと、
抽出されたピクセル差成分と１組の前もってストアされたピクセル成分とから新しい組のピクセル成分を計算するアキュムレータとを含む、データフォーマッタ。

【請求項３】　前記新しい組のピクセル成分をフォーマット化する第２のフォーマッタを含む、請求項２に記載のデータフォーマッタ。

【請求項４】　レンダリングバスに結合される記憶装置のためのデータフォーマッタであって、
ピクセルデータから複数のピクセル成分を抽出する第１のフォーマッタを含み、各抽出されたピクセル成分はピクセルに対するウインドウ識別データフィールドに対応しており、前記データフォーマッタはさらに
前記第１のフォーマッタに結合され、複数の選択可能なオペレーションのモードをストアするメモリユニットと、
前記メモリユニットに結合され、前記メモリユニットにストアされる１つ以上の選択可能なオペレーションのモードに従って新しいピクセルデータをフォーマット化する第２のフォーマッタとを含む、データフォーマッタ。

【請求項５】　ピクセルデータから複数のピクセル成分を抽出し、抽出されたピクセル成分を複数の群のピクセルデータに組合せる第１のフォーマッタと、
前記第１のフォーマッタに結合され、前記複数の群のピクセルデータを組立てる第２のフォーマッタとを含む、記憶装置のためのデータフォーマッタ。

2

【請求項６】　ピクセルデータから前記ピクセルデータの複数の群を抽出する第１のフォーマッタと、
前記第１のフォーマッタに結合され、前記ピクセルデータの前記群から複数のピクセル成分を抽出する第２のフォーマッタとを含む、記憶装置のためのデータフォーマッタ。

【請求項７】　記憶装置であって
データをストアするためのメモリアレイを含み、前記メモリアレイは複数のメモリバンクを含み、各メモリバンクは複数のメモリセルと、前記複数のメモリセルにアクセスするように結合される複数のセンスアンプとを含んでおり、前記記憶装置はさらに前記メモリアレイの前記センスアンプに結合されるバッファを含み、前記バッファは複数のキャッシュラインを含んでおり、前記記憶装置はさらに前記バッファに結合される複数の算術論理演算装置と、
前記複数の算術論理演算装置に結合され、前記記憶装置の外部にあるレンダリングバスに結合される入力データフォーマッタと、
前記バッファおよび外部レンダリングバスに結合される出力データフォーマッタと、
前記複数の算術論理演算装置および前記バッファに結合される書込データフォーマッタと、
前記バッファおよび前記複数の算術論理演算装置に結合される読出データフォーマッタと、
前記メモリアレイ、前記バッファ、前記複数の算術論理演算装置、前記入力データフォーマッタ、前記出力データフォーマッタ、前記書込データフォーマッタ、前記読出データフォーマッタおよび、前記記憶装置の外部にあるアドレスおよび制御バスに結合されるアドレスおよび制御入力バスとを含む、記憶装置。

【請求項８】　前記入力データフォーマッタは、
前記レンダリングバスを介して前記入力データフォーマッタへ送信される１組のピクセル差成分を抽出する第１のフォーマッタと、
抽出されたピクセル差成分と１組の前もってストアされたピクセル成分とから新しい組のピクセル成分を計算するアキュムレータとを含む、請求項７に記載の記憶装置。

【請求項９】　前記新しい組のピクセル成分をフォーマット化する第２のフォーマッタを含む、請求項８に記載の記憶装置。

【請求項１０】　メモリバンクオペレーションを特定するメモリバンクコマンドを受信するステップと、
グローバルバスオペレーションを特定するグローバルバスコマンドを受信するステップと、
算術論理演算装置オペレーションを特定するデータ処理コマンドを受信するステップと、
前記メモリバンクコマンドに従って特定されたメモリバンクオペレーションを実行し、前記グローバルバスコマ

ンドに従って特定されたグローバルバスオペレーションを実行し、前記データ処理コマンドに従って特定された算術論理演算装置オペレーションを実行するステップとを含む、少なくとも１つの記憶装置内のデータにアクセスする方法。

【請求項１１】　メモリのキャッシュライン内にストアされる複数のバイトのデータの各々に対応するデータを書込イネーブルするか、または書込ディスエーブルするかを制御するために少なくとも１つのバイトマスクレジスタを設定するステップと、

メモリの前記キャッシュライン内にストアされる前記複数のバイトのデータの各々の成分に対応するデータを書込イネーブルするか、または書込ディスエーブルするかを制御するために少なくとも１つのプレーンマスクレジスタを設定するステップと、

前記少なくとも１つのバイトマスクレジスタおよび前記少なくとも１つのプレーンマスクレジスタの書込イネーブル設定により制御されるように、特定されたメモリバンクおよびコラムアドレスに結合される１組の前もって活性化されたセンスアンプにキャッシュラインメモリの内容を書込むステップとを含む、記憶装置内のデータの領域をクリアする方法。

【請求項１２】　キャッシュラインメモリの内容を書込む前記ステップを、特定された数のメモリバンクおよびコラムアドレスに結合される特定された数の組の前もって活性化されたセンスアンプに対して、特定された回数だけ繰返すステップを含む、請求項１１に記載の方法。

【請求項１３】　前記少なくとも１つのバイトマスクレジスタおよび前記少なくとも１つのプレーンマスクレジスタの前記書込イネーブル設定により制御されるように、前記キャッシュラインメモリの前記内容を書込む前記ステップは、特定されたメモリバンクおよびコラムアドレスに結合される複数の組の前もって活性化されたセンスアンプに対して同時に実行される、請求項１１に記載の方法。

【請求項１４】　コントローラからメモリへ送信されるデータを圧縮する方法であって、

前記コントローラおよび前記メモリの両方に、複数の古いピクセルに対応する古いピクセルデータをストアするステップを含み、前記古いピクセルデータは前記古いピクセルデータにより表わされる古いピクセルの各々に対する１組の古いピクセル成分を含んでおり、前記方法はさらに前記コントローラにおいて、複数の新しいピクセルのための新しいピクセルデータを計算するステップを含み、前記新しいピクセルデータは前記新しいピクセルデータにより表わされる新しいピクセルの各々に対する１組の新しいピクセル成分を含んでおり、前記方法はさらに前記コントローラにおいて、前記新しいピクセルデータのある特定のピクセル成分と前記古いピクセルデータのある特定のピクセル成分との差を計算するステップと、

前記コントローラにおいて、前記新しいピクセルデータのある特定のピクセル成分と前記古いピクセルデータのある特定のピクセル成分との前もって計算された差の差を計算するステップと、

前記コントローラにおいて、計算された差と、差の計算された差とに基づいて圧縮フォーマットを決定するステップと、

前記コントローラにおいて前記計算された差と前記差の計算された差とを決定された圧縮フォーマットにフォーマット化することにより前記新しいピクセルデータを圧縮するステップと、

圧縮された新しいピクセルデータを前記コントローラから前記メモリへ送信するステップと、

前記圧縮された新しいピクセルデータを前記メモリ内で伸長するステップとを含む、方法。

【請求項１５】　前記コントローラにおいて複数の新しいピクセルのための新しいピクセルデータを計算する前記ステップは、差の値を、これらが計算された元のピクセルデータの幅に符号拡張するステップを含み、前記新しいピクセルデータは前記新しいピクセルデータにより表わされる新しいピクセルの各々に対する１組の新しいピクセル成分を含む、請求項１４に記載の方法。

【請求項１６】　前記圧縮された新しいピクセルデータを前記メモリ内で伸長する前記ステップは、

複数のデータフォーマットから前記計算された差と前記差の計算された差とを抽出するステップと、

抽出された差および差の差を、対応する古いピクセル成分に対応するデータ幅に符号拡張するステップと、

符号拡張された差および差の差をストアされた古いピクセル成分に加算することによって新しいピクセル成分を再生するステップと、

前記新しいピクセルデータをフォーマット化するステップとを含む、請求項１４に記載の方法。

【請求項１７】　ピクセルデータを含むメモリのキャッシュラインから複数のウインドウ識別ピクセル成分を抽出するステップと、

ウインドウ識別フィールドが抽出された各ピクセルに対して、色ピクセルデータのどの部分とオーバレイピクセルデータのどの部分とを前記コントローラに送信するかを決定するステップと、

抽出された複数のウインドウ識別ピクセル成分、色ピクセルデータの決定された部分およびオーバレイピクセルデータの決定された部分を前記コントローラに送信するステップとを含む、メモリからコントローラへ送信されるデータをフォーマット化する方法。

【請求項１８】　少なくとも１つの算術論理演算装置から複数のピクセル成分を受信するステップと、

前記複数のピクセル成分をピクセルデータの群にパックするステップと、

前記ピクセルデータの群を１ブロックのデータに組立てるステップとを含む、記憶装置においてデータをフォーマット化する方法。

【請求項１９】　少なくとも１つの算術論理演算装置から複数のピクセル成分を受ける前記ステップはピクセル色成分を受信するステップと、
ピクセルデプス成分を受信するステップと、
ピクセルステンシル成分を受信するステップとを含む、請求項１８に記載の方法。

【請求項２０】　バッファからピクセルデータのブロックを受信するステップと、
前記ピクセルデータのブロックから複数の群の選択されたピクセルデータを抽出するステップと、
前記複数の群の選択されたピクセルデータから複数の選択されたピクセル成分を抽出するステップとを含む、記憶装置においてデータをフォーマット化する方法。

【請求項２１】　コンピュータによる動作のためのグラフィックスシステムであって、
レンダリングコントローラと、
前記レンダリングコントローラと、前記グラフィックスシステムを動作させるのに用いられる前記コンピュータとに結合されるインタフェースと、
記憶装置と、
前記レンダリングコントローラおよび前記記憶装置に結合されるレンダリングバスと、
前記レンダリングコントローラおよび前記記憶装置に結合されるアドレスおよび制御バスとを含む、グラフィックスシステム。

【請求項２２】　ビデオ出力チャネルにより前記レンダリングコントローラに結合されるビデオ出力回路を含む、請求項２１に記載のグラフィックスシステム。

【請求項２３】　前記記憶装置は、前記レンダリングバスと前記アドレスおよび制御バスとに結合される複数のメモリチップを含む、請求項２１に記載のグラフィックスシステム。

【請求項２４】　レンダリングバスとアドレスおよび制御バスとの複数の対を含む、請求項２３に記載のグラフィックスシステム。

【請求項２５】　最大数のメモリチップが、レンダリングバスとアドレスおよび制御バスとの各対に結合される、請求項２４に記載のグラフィックスシステム。

【請求項２６】　レンダリングバスとアドレスおよび制御バスとの各対に結合されるメモリチップの前記最大数は４である、請求項２５に記載のグラフィックスシステム。

【請求項２７】　前記レンダリングバスは同時双方向送受信を含み、データを同時に前記レンダリングコントローラから前記記憶装置に送信し前記記憶装置から前記レンダリングコントローラへ送信することを可能にする、請求項２１に記載のグラフィックスシステム。



【請求項２８】　前記記憶装置は
データをストアするためのメモリアレイと、
前記メモリアレイに結合されるバッファと、
前記バッファに結合され、前記記憶装置の外部にあるレンダリングバスに結合される複数の算術論理演算装置と、
外部レンダリングバスおよび前記複数の算術論理演算装置に結合される入力データフォーマッタと、
前記バッファおよび前記外部レンダリングバスに結合される出力データフォーマッタと、
前記複数の算術論理演算装置および前記バッファに結合される書込データフォーマッタと、
前記バッファおよび前記複数の算術論理演算装置に結合される読出データフォーマッタと、
前記メモリアレイ、前記バッファ、前記複数の算術論理演算装置、前記入力データフォーマッタ、前記出力データフォーマッタ、前記書込データフォーマッタ、前記読出データフォーマッタおよび、前記記憶装置の外部にあるアドレスおよび制御バスに結合されるアドレスおよび制御入力バスとを含む、請求項２１に記載のグラフィックスシステム。

【請求項２９】　前記メモリアレイは複数のメモリバンクを含み、各メモリバンクは複数のメモリページを含み、各メモリページは複数のメモリラインを含み、各メモリラインは複数のメモリセルを含み、各メモリセルは１ビットのデータをストアすることができる、請求項２８に記載のグラフィックスシステム。

【請求項３０】　前記複数の算術論理演算装置は、複数のラスタオペレーションユニット、複数のブレンドユニット、複数のデプスユニット、複数のステンシルユニットおよび複数のウインドウ識別ユニットを含み、複数のピクセルに対応するデータを同時に処理するようにする、請求項２８に記載のグラフィックスシステム。

【請求項３１】　前記入力データフォーマッタは、第２のフォーマッタに結合されるアキュムレータに結合される第１のフォーマッタを含む、請求項２８に記載のグラフィックスシステム。

【請求項３２】　前記出力データフォーマッタは、複数のウインドウ識別データ抽出器と、複数のオーバレイデータセレクタと、複数の色データセレクタとに結合されるモードレジスタを含む、請求項２８に記載のグラフィックスシステム。

【請求項３３】　前記読出データフォーマッタはモードレジスタおよびマスクレジスタを含み、前記レジスタの各々は複数のデータ抽出器および複数のデータアンパッカーに結合される、請求項２８に記載のグラフィックスシステム。

【請求項３４】　前記書込データフォーマッタはモードレジスタおよびマスクレジスタを含み、前記レジスタの各々は複数の色パックユニットと、複数のデプスパック

ユニットと、複数のエクストラパックユニットとに結合される、請求項28に記載のグラフィックスシステム。

【請求項３５】　前記アドレスおよび制御入力バスは、複数のアドレスおよび制御チャネルに結合されるデコーダを含む、請求項２８に記載のグラフィックスシステム。

【請求項３６】　コンピュータによる動作のためのグラフィックスシステムであって、

コントローラに結合される記憶装置のためのデータフォーマッタを含み、前記データフォーマッタは前記コントローラから前記データフォーマッタへ送信される１つ以上の信号から１組のピクセル差成分を抽出する第１のフォーマッタと、

抽出されたピクセル差成分と１組の前もってストアされたピクセル成分とから新しい組のピクセル成分を計算するアキュムレータとを含む、グラフィックスシステム。

【請求項３７】　前記新しい組のピクセル成分をフォーマット化する第２のフォーマッタを含む、請求項３６に記載のグラフィックスシステム。

【請求項３８】　コンピュータによる動作のためのグラフィックスシステムであって、

レンダリングバスに結合される記憶装置のためのデータフォーマッタを含み、前記データフォーマッタはピクセルデータから複数のピクセル成分を抽出する第１のフォーマッタを含み、各抽出されたピクセル成分はピクセルに対するウインドウ識別データフィールドに対応しており、前記データフォーマッタはさらに前記第１のフォーマッタに結合され、複数の選択可能なオペレーションのモードをストアするメモリユニットと、

前記メモリユニットに結合され、前記メモリユニットにストアされる１つ以上の前記選択可能なオペレーションのモードに従って新しいピクセルデータをフォーマット化する第２のフォーマッタとを含む、グラフィックスシステム。

【請求項３９】　コンピュータによる動作のためのグラフィックスシステムであって、

記憶装置のためのデータフォーマッタを含み、前記データフォーマッタはピクセルデータから複数のピクセル成分を抽出して、抽出されたピクセル成分を複数の群のピクセルデータに組合せる第１のフォーマッタと、

前記第１のフォーマッタに結合され、前記複数の群のピクセルデータを組立てる第２のフォーマッタとを含む、グラフィックスシステム。

【請求項４０】　コンピュータによる動作のためのグラフィックスシステムであって、

記憶装置のためのデータフォーマッタを含み、前記データフォーマッタはピクセルデータから前記ピクセルデータの複数の群を抽出する第１のフォーマッタと、

前記第１のフォーマッタに結合され、前記ピクセルデータの前記群から複数のピクセル成分を抽出する第２のフォーマッタとを含む、グラフィックスシステム。

【請求項４１】　コンピュータによる動作のためのグラフィックスシステムであって、

データをストアするためのメモリアレイを含み、前記メモリアレイは複数のメモリバンクを含み、各メモリバンクは複数のメモリセルと、前記複数のメモリセルにアクセスするよう結合される複数のセンスアンプとを含んでおり、前記グラフィックスシステムはさらに前記メモリアレイの前記センスアンプに結合されるバッファを含み、前記バッファは複数のキャッシュラインを含み、前記グラフィックスシステムはさらにピクセルバッファに結合され、記憶装置の外部にあるレンダリングバスに結合される複数の算術論理演算装置と、

外部レンダリングバスおよび前記複数の算術論理演算装置に結合される入力データフォーマッタと、

前記バッファおよび前記外部レンダリングバスに結合される出力データフォーマッタと、

前記複数の算術論理演算装置および前記バッファに結合される書込データフォーマッタと、

前記バッファおよび前記複数の算術論理演算装置に結合される読出データフォーマッタと、

前記メモリアレイ、前記バッファ、前記複数の算術論理演算装置、前記入力データフォーマッタ、前記出力データフォーマッタ、前記書込データフォーマッタ、前記読出データフォーマッタおよび、前記記憶装置の外部にあるアドレスおよび制御バスに結合されるアドレスおよび制御入力バスとを含む、グラフィックスシステム。

【請求項４２】　前記入力データフォーマッタは前記レンダリングバスを介して前記入力データフォーマッタへ送信される１組のピクセル差成分を抽出する第１のフォーマッタと、

抽出されたピクセル差成分と１組の前もってストアされたピクセル成分とから新しい組のピクセル成分を計算するアキュムレータとを含む、請求項４１に記載のグラフィックスシステム。

【請求項４３】　前記新しい組のピクセル成分をフォーマット化する第２のフォーマッタを含む、請求項４２に記載のグラフィックスシステム。

【請求項４４】　コンピュータに関連してグラフィックスシステムを動作させる方法であって、

レンダリングコントローラから記憶装置へレンダリングバスを介して入力データを送信するステップと、

前記レンダリングコントローラから前記記憶装置へアドレスおよび制御バスを介してアドレスおよび制御データを送信するステップと、

前記入力データをフォーマット化するステップと、

フォーマット化された入力データを処理して処理されたデータを得るステップと、

前記処理されたデータをフォーマット化するステップと、

フォーマット化された処理されたデータをメモリにストアするステップと、
前記処理されたデータを前記メモリからアクセスするステップと、
アクセスされた処理されたデータのいくつかをビデオ出力回路への送信のためにフォーマット化し、かつそのようにフォーマット化された処理されたデータを前記記憶装置から前記レンダリングバスを介して送信するステップと、
再処理すべき前記アクセスされた処理されたデータのいくつかをフォーマット化し、かつそのようにフォーマット化された前記再処理すべき処理されたデータを送信するステップとを含む、方法。

【請求項４５】　前記レンダリングコントローラから前記記憶装置へアドレスおよび制御バスを介してアドレスおよび制御データを送信する前記ステップは、
メモリバンクオペレーションを特定するメモリバンクコマンドを送信するステップと、
グローバルバスオペレーションを特定するグローバルバスコマンドを送信するステップと、
ピクセル算術論理演算装置オペレーションを特定するデータ処理コマンドを送信するステップとを含み、
フォーマット化された入力データを処理して処理されたデータを得る前記ステップと、フォーマット化された処理されたデータをメモリにストアする前記ステップと、
前記処理されたデータを前記メモリからアクセスする前記ステップとは前記メモリバンクコマンドに従って特定されたメモリバンクオペレーションを実行し、前記グローバルバスコマンドに従って特定されたグローバルバスオペレーションを実行し、前記データ処理コマンドに従って特定されたピクセル算術論理演算装置オペレーションを実行するステップを含む、請求項４４に記載の方法。

【請求項４６】　フォーマット化された処理されたデータをメモリにストアする前記ステップはメモリのキャッシュライン内にストアされる複数のバイトのデータの各々に対応するデータを書込イネーブルするか、または書込ディスエーブルするかを制御するために少なくとも１つのバイトマスクレジスタを設定するステップと、
メモリの前記キャッシュライン内にストアされる前記複数のバイトのデータの各々の成分に対応するデータを書込イネーブルするか、または書込ディスエーブルするかどうかを制御するために少なくとも１つのプレーンマスクレジスタを設定するステップと、
前記少なくとも１つのバイトマスクレジスタおよび前記少なくとも１つのプレーンマスクレジスタの書込イネーブル設定により制御されるように、特定されたメモリバンクおよびコラムアドレスに結合される１組の前もって活性化されたセンスアンプにキャッシュラインメモリの内容を書込むステップとを含む、請求項４４に記載の方法。

【請求項４７】　キャッシュラインメモリの内容を書込む前記ステップを、特定された数のメモリバンクおよびコラムアドレスに結合される特定された数の組の前もって活性化されたセンスアンプに対して、特定された回数だけ繰返すステップを含む、請求項４６に記載の方法。

【請求項４８】　前記少なくとも１つのバイトマスクレジスタおよび前記少なくとも１つのプレーンマスクレジスタの前記書込イネーブル設定により制御されるように、前記キャッシュラインメモリの前記内容を書込む前記ステップは、特定されたメモリバンクおよびコラムアドレスに結合される複数の組の前もって活性化されたセンスアンプに対して同時に実行される、請求項４６に記載の方法。

【請求項４９】　レンダリングコントローラから記憶装置へレンダリングバスを介して入力データを送信する前記ステップと、前記入力データをフォーマット化する前記ステップとは、
前記レンダリングコントローラおよび前記メモリの両方に、複数の古いピクセルに対応する古いピクセルデータをストアするステップを含み、前記古いピクセルデータは前記古いピクセルデータにより表わされる古いピクセルの各々に対する１組の古いピクセル成分を含んでおり、前記ステップはさらに前記コントローラにおいて、複数の新しいピクセルのための新しいピクセルデータを計算するステップを含み、前記新しいピクセルデータは前記新しいピクセルデータにより表わされる新しいピクセルの各々に対する１組の新しいピクセル成分を含んでおり、前記ステップはさらに前記コントローラにおいて、前記新しいピクセルデータのある特定のピクセル成分と前記古いピクセルデータのある特定のピクセル成分との差を計算するステップと、
前記コントローラにおいて、前記新しいピクセルデータのある特定のピクセル成分と前記古いピクセルデータのある特定のピクセル成分との前もって計算された差の差を計算するステップと、
前記コントローラにおいて、計算された差と、差の計算された差とに基づいて圧縮フォーマットを決定するステップと、
前記コントローラにおいて前記計算された差と前記差の前記計算された差とを決定された圧縮フォーマットにフォーマット化することにより前記新しいピクセルデータを圧縮するステップと、
圧縮された新しいピクセルデータを前記コントローラから前記メモリへ送信するステップと、
前記圧縮された新しいピクセルデータを前記メモリ内で伸長するステップとを含む、請求項４４に記載の方法。

【請求項５０】　前記コントローラにおいて複数の新しいピクセルのための新しいピクセルデータを計算する前記ステップは、差の値を、これらが計算された元のデー

タの幅に符号拡張するステップを含み、前記新しいピクセルデータは前記新しいピクセルデータにより表わされる新しいピクセルの各々に対する1組の新しいピクセル成分を含む、請求項４９に記載の方法。

【請求項５１】　前記圧縮された新しいピクセルデータを前記メモリ内で伸長する前記ステップは、

複数のデータフォーマットから前記計算された差と前記差の計算された差とを抽出するステップと、

抽出された差および差の差を、対応する古いピクセル成分に対応するデータ幅に符号拡張するステップと、

符号拡張された差および差の差をストアされた古いピクセル成分に加算することによって新しいピクセル成分を再生するステップと、

前記新しいピクセルデータをフォーマット化するステップとを含む、請求項４９に記載の方法。

【請求項５２】　アクセスされたピクセルデータのいくつかをビデオ出力回路への送信のためにフォーマット化し、かつそのようにフォーマット化されたデータを前記レンダリングバスを介して送信する前記ステップは、

ピクセルデータを含むメモリのキャッシュラインから複数のウインドウ識別ピクセル成分を抽出するステップと、

ウインドウ識別フィールドが抽出された各ピクセルに対して、色データのどの部分とオーバレイデータのどの部分とを前記コントローラに送信するかを決定するステップと、

抽出された複数のウインドウ識別ピクセル成分、色データの決定された部分およびオーバレイデータの決定された部分を前記コントローラに送信するステップとを含む、請求項４４に記載の方法。

【請求項５３】　前記処理されたデータをフォーマット化する前記ステップは、

少なくとも１つの算術論理演算装置から複数のピクセル成分を受信するステップと、

前記複数のピクセル成分をピクセルデータの群にパックするステップと、

前記ピクセルデータの群を１ブロックのピクセルデータに組立てるステップとを含む、請求項４４に記載の方法。

【発明の詳細な説明】

【０００１】

【発明の分野】開示される発明は一般的にコンピュータシステムに関する。より特定的には、この発明はダイナミックランダムアクセスメモリ(DRAM)フレームバッファ装置と、その装置に基づく、増速された二次元および三次元グラフィックスレンダリングオペレーションを実行するためのアーキテクチャを提供するシステムとに関する。

【０００２】

【背景】デュアルピクセル３DRAMチップおよびグラフィックス処理システムは、高性能で高容量のフレームバッファを実現するのに用いられる。開示されるデュアルピクセル３DRAMチップおよびデュアルピクセル３DRAMチップに基づくグラフィックス処理システムのある局面は、１９９６年８月６日にデーリング(Deering)他に発行された米国特許第５，５４４，３０６号に開示されており、この特許はここに完全に提示されているかのごとくにその全体においてこの開示に引用により援用される。

【０００３】開示される発明は外部DRAMフレームバッファの使用に替るものを提示する。短期間の性能上の目的を満たすためには、組込みDRAMをフレームバッファメモリに用いたくなる。なぜなら、かなりの量の論理を実現するための表面積を残しつつ、ダイ上に４から８メガビットを組込むことが可能だろうからである。しかしながら、同じ時間フレームにおいて、グラフィックス指向の計算機プロダクトはフレームバッファメモリの１０から８０メガビットを要する。このため、グラフィックス処理計算機システムの要件を満たすようなフレームバッファを実現するためには、２個から１０個の組込みDRAM装置が必要となるであろう。そのようなフレームバッファのフィル速度は非常に高速だろうが、計算機市場の大部分にとってはそのコストは高すぎるであろう。

【０００４】二次元および三次元グラフィックイメージを処理する上で、フィル速度と一致させるためテクスチャマッピングを増速させなければならない。しかしながら、上述の分散されたフレームバッファではこれを効率よく行なうことが難しくなる。テクスチャデータを分配する最も簡単なやり方は、各装置がすべてのものの複製を持つことだが、この方法は非常に効率の悪い組込みDRAMビットの用い方である。別のアプローチでは、データが複製されない態様でテクスチャデータをフレームバッファを含む装置の間で分配する。この選択肢では、組込みDRAMビットが有効に用いられるが、装置間での大量のテクスチャデータのルーティングも必要となる。

【０００５】組込みDRAMを用いて単一の装置上でテクスチャキャッシュを実現する方がより実用的であるかもしれない。テクスチャデータはシステムメモリからページインしなければならないが、これはテクスチャデータが圧縮されている方がより効率よく働くであろう。

【０００６】組込みDRAMは、セットアップユニットとラスタライザとの間の単純なFIFOを実現するのに用いることもできる。このFIFOにより、大きな三角形がレンダリングされる間、ジオメトリおよびセットアップ処理を続けることが可能となる。これはまた、システムメモリからのテクスチャデータのページングおよび伸長のレイテンシを緩和するのに用いることもできる。

【０００７】Talisman、Pixel FlowまたはOak's WARP5などの領域ベースのレンダリングアーキテクチャでは、レンダリングコントローラ上でフレームバッファの小さな部分をレンダリングし、次に最終の色値を外部DRAMに

転送する。コントローラはさらに次の領域をレンダリングし、さらにその次の領域へと、フレーム全体にわたってレンダリングし続ける。

【０００８】隠面消去およびアンチエイリアシングに用いられる帯域幅のすべてがレンダリングコントローラに完全に残るため、フィル速度は外部帯域幅により制限されない。隠面消去およびアンチエイリアシングに用いられる記憶域のすべてはフレームバッファの小さい部分に対してのみ実現されるだけでよく、したがってレンダリングコントローラ上に置いておくことができる。

【０００９】領域ベースのレンダリングの大きな欠点は、レンダリングを開始できるようになるまでに、各フレームごとにすべてのジオメトリを領域に分類してどこかにストアする必要があることである。この要件のため、一般的にフレームごとにレンダリングできるジオメトリの量に上限が課せられる。この制限は数多くの応用において許容できないものである。いくつかの領域ベースのレンダリングアーキテクチャは、大量のジオメトリを与えられた際にも外部DRAMから、および外部DRAMへ領域のためのデプス値および色値を書込むことによりある程度正確に機能できる。しかしながら、こうした実装では領域ベースのレンダリングの欠点のすべてが保たれたまま利点のすべてが損われる。

【００１０】機械設計CADおよび他のコンテンツ作成応用ではジオメトリの複雑さに対する制限は許容できない。このような応用ではジオメトリの複雑さをフレーム速度とスムーズにトレードオフする能力が必要となる。このように、組込みDRAMのアプローチも領域ベースのレンダリングのアプローチも、現在のグラフィックス処理応用の性能の要求および実際のコスト制約を満たす十分な解決策を提供するものではない。

【００１１】フレームバッファ設計に関する別の懸念は、シングルポートのフレームバッファメモリとデュアルポートのフレームバッファメモリとの性能トレードオフである。デュアルポートフレームバッファは専用ディスプレイポートを有しており、これによりレンダリングポートがより長い時間をレンダリングに費やすことができる。典型的に、ビデオランダムアクセスメモリ(VRAM)チップを含むデュアルポートフレームバッファは、そのビデオバッファがかなり大きいため、そのフィル速度の約１％から２％しかビデオ転送オペレーションのために失わない。FBRAMチップ(3D-RAM™ チップとも称される)を含むフレームバッファは、ビデオバッファがより小さいため、そのフィル速度の約５％から１０％だけビデオ転送オペレーションのため失う。

【００１２】シングルポートメモリは、ポートが高速双方向送受信をサポートするのでない限り、表示するためにピクセルデータを読出している際にはレンダリングすることができない。双方のレンダリングポート帯域幅が同じであるものとしてシングルポートメモリとデュアルポートメモリとを比較すると、デュアルポートメモリの方がフィル速度がより速く、コストもより高い。シングルポートの帯域幅がデュアルポートの帯域幅の合計に等しいものとしてシングルポートメモリとデュアルポートメモリとを比較すると、シングルポートメモリの方がより効率がよいため、おそらくシングルポートメモリのフィル速度の方がデュアルポートメモリよりも高速であろう。すなわち、高帯域幅入力／出力(I/O)能力の出現により現在、帯域幅の制限が緩和されつつある限りにおいて、シングルポートメモリアーキテクチャの方がより効率のよいフレームバッファ性能を期待できる。



【００１３】デュアルポートメモリではフレームバッファへのピクセルのフローがより滑らかである。シングルポートメモリはディスプレイデータのバーストを読出している間は定期的にレンダリングに利用できなくなる。レンダリングコントローラはシングルポートメモリとインターフェイスする際にはピクセルフローを滑らかにするのにより大きなピクセルFIFOを必要とする。より低コストのシステムでは、そのようなディスプレイバーストの間、レンダラーはアイドル状態となるかもしれない。

【００１４】シングルポートメモリは、ダイ面積、ピン、パッケージング、テストおよび電力消費が小さいため費用がより安い。シングルポートメモリは同じ大きさのデュアルポートメモリと比べて、ビット当りの記憶コストがかなり低い。ビット当りのコストが低ければ、表示できないデータをフレームバッファにストアしてもそれほど問題にならない。

【００１５】デュアルポートメモリはディスプレイ帯域幅が固定されている。必要となるディスプレイ帯域幅がより低ければ帯域幅は無駄になる。必要とされるディスプレイ帯域幅がより高ければ、そのメモリはそのディスプレイ要件に適していないことになる。シングルポートメモリには、レンダリング帯域幅とディスプレイ帯域幅とをトレードオフする柔軟性がある。緊急時には、シングルポートメモリは実際に非常に高いディスプレイ帯域幅を提供することができる。

【００１６】デュアルポートメモリの専用ディスプレイポートは水平および垂直帰線消去期間の間は用いられないが、このことはディスプレイポートが約２０パーセントの間はアイドル状態であることを意味する。

【００１７】デュアルポートメモリでは、スクリーンへのピクセルおよびブロックのマッピングを固定することが要求される。比較して、シングルポートメモリではかなりの柔軟性を持ってピクセルおよびブロックをスクリーンへマッピングできる。

【００１８】デュアルポートフレームバッファメモリは、レンダリングポートおよびディスプレイポートが異なるチップに接続されている場合にしか意味をなさない。両方のポートが同じチップに接続されている場合には、帯域幅の等しいシングルポートメモリの方が上に挙

げた理由からより有効であろう。

【００１９】シングルポートメモリでは、フレームバッファのビット当りの記憶コストがより安く、レンダリングチップとディスプレイチップとを単一の装置に合併させることができるため、より低コストのシステムを製造できる。

【００２０】このように、シングルポートメモリではデュアルポートメモリで設計できるものより低コストで低価格帯のシステムを設計することが可能になる。ビット当りの記憶コストはシングルポートメモリではかなり低いので高解像度／高ピクセルデプス設計において材料コストはかなり低くなる。シングルポートメモリはより柔軟であるため、より広範囲のプロダクト能力を提供する設計をもたらす。

【００２１】

【発明の概要】この発明は、コンピュータグラフィックスシステムにおいて二次元および三次元イメージの増速されたレンダリングをもたらすシングルポートフレームバッファアクセスメモリ(デュアルピクセル３DRAM)チップに向けられる。

【００２２】デュアルピクセル３DRAMチップは、レンダリングバスを介してレンダリングコントローラによりアクセスされるシングルポートの高速メモリを特徴とする。デュアルピクセル３DRAMチップは、DRAMアレイ、SRAMピクセルバッファ、少なくとも１つのピクセル算術論理演算装置(ALU)およびグローバルバスを含む。また、デュアルピクセル３DRAMチップはいくつかのデータバスおよびデータフォーマッタを含み、これらはグラフィックスデータがデュアルピクセル３DRAMチップ内で処理されたり、更新されたり、そのチップから送信されたり、そのチップ内にストアされたりする際にそのグラフィックスデータを割振り、フォーマット化する。

【００２３】この発明の第１の局面において、デュアルピクセル３DRAMチップは、８ビットピクセルから最大５１２ビットピクセルまでの範囲のさまざまなピクセルサイズおよびフォーマットを処理するよう設定可能である。デュアルピクセル３DRAMチップはこれらの能力を実現するため、新規のプロトコルおよびデータパッキング方式を特徴とする。

【００２４】この発明の別の局面において、デュアルピクセル３DRAMチップは、チップ上での２サイクルおよび３サイクルのピクセルALUオペレーションの両方を可能にする、レンダリングバスを介する可変入力および出力データレートをサポートする。

【００２５】この発明の別の局面において、デュアルピクセル３DRAMチップは同時にオペレーションごとに２つの別個のピクセルまたはサンプルを処理する。

【００２６】この発明の別の局面において、デュアルピクセル３DRAMチップは、チップとレンダリングコントローラとの間でより高速のフィル速度およびスループットを可能にするデータ圧縮能力を特徴とする。



【００２７】この発明の別の局面において、デュアルピクセル３DRAMチップは、アンチエイリアシングされたポリゴンをレンダリングするため新規のデルタＺアルゴリズムを用いるマルチサンプリング方式を用いる。

【００２８】この発明の別の局面において、デュアルピクセル３DRAMチップは、レンダリングコントローラとチップとの間のアドレスおよび制御バス上の帯域幅要件を最小にするためチップ上にDRAMバンクおよびコラムアドレスを保持するための新規の方式を用いる。

【００２９】この発明の別の局面において、デュアルピクセル３DRAMチップは、帯域幅のバランスを取りチップのオペレーション効率を最適化するような、チップ内部のデータバスの幅とデータ転送速度との関係を含む。

【００３０】この発明の別の局面において、デュアルピクセル３DRAMチップは多倍精度ピクセルブレンドオペレーションを行ない、どんなビット幅の入力もブレンドされるようにする。

【００３１】この発明の別の局面において、デュアルピクセル３DRAMチップはアドレスおよび制御情報を３つの別個の組の信号に分割し、３つの別個の組の信号は同時に送信されてチップ上でのDRAMバンクオペレーション、グローバルバスオペレーションおよびピクセルALUオペレーションを制御する。

【００３２】この発明の別の局面において、デュアルピクセル３DRAMチップは、DRAMバンクコラムデコーダとセンスアンプとの間の複数のバスに書込み、この結果フレームバッファのクリア速度を４倍以上に増大させるFlash Line(フラッシュライン)オペレーションを特徴とする。

【００３３】この発明の別の局面において、デュアルピクセル３DRAMチップは、DRAMアレイとSRAMピクセルバッファとの間の双方向グローバルバスにより、異なるレベルのキャッシュ間でのデータの同時転送を可能にする、新規のオペレーションであるChange Cache Line(キャッシュライン変更)を特徴とする。

【００３４】この発明の別の局面において、デュアルピクセル３DRAMチップは、ページプリチャージオペレーションとページバンクアクセスオペレーションとを１つのオペレーションに組合せた、Change Page bank(ページバンク変更)オペレーションを特徴とする。

【００３５】この発明の別の局面において、デュアルピクセル３DRAMチップは、ある特定のレジスタのデータまたは内容がチップ上のある特定のバスを介してブロードキャストされるピクセルALUオペレーションを特徴とする。

【００３６】この発明の別の局面において、デュアルピクセル３DRAMチップは、SRAMピクセルバッファからシングルでも、デュアルピクセルフォーマットでも、ピクセルデータを読出すための革新的な手段を特徴とする。

【００３７】オペレーション、構造、部品の組立および組合せのさまざまな新規の詳細を含むこの発明の上述および他の特徴を以下に添付の図面に関連してより特定的に説明する。この発明の特定の実施例はここに例示としてのみ開示されており、クレームされる発明に限定を課するものではないことが理解されるであろう。この発明の原理および特徴はこの発明の範囲から逸脱することなく数多くのさまざまな実施例において用いられ得る。

【００３８】

【詳細な説明】以下に、現在発明者により企図されるこの発明を実施するためのベストモードに基づいてこの発明を詳細に説明する。以下、図面の簡単な説明において説明される添付の図面を参照するが、図面すべてを通じて要素に一貫した番号を振っている。この開示を通じて、当業者には既知のVerilogハードウェア記述言語（ＶＨＤＬ）構文法で表現されるデュアルピクセル３DRAMチップのさまざまな機能的な側面を記述する。

【００３９】目次

1.0　アーキテクチャ
1.1 ピクセルALU
1.1.1　ROP/ブレンドユニット
1.1.2　デプスユニット
1.1.3　ステンシルユニット
1.1.4　ウィンドウIDユニット
1.2 SRAMピクセルバッファ
1.3 メモリ構成
1.4 ピン構成
1.5 プロトコル
1.5.1　DRAMバンクオペレーション
1.5.2　グローバルバスオペレーション
1.5.3　ピクセルALUオペレーション
1.6 オペレーションタイミング
1.7 レジスタ
1.7.1　識別
1.7.2　FeatureEnable
1.7.3　PixelConfig
1.7.4　StencilDepthConfig
1.7.5　ColorOP[0]
1.7.6　ColorOP[1]
1.7.7　ConstantColor
1.7.8　Byte Mask[1:0]
1.7.9　Plane Mask[7:0]
1.7.10　ColorWIDLUT[3:0]
1.7.11　OverlayWIDLUT[3:0]
1.7.12　DisplayConfig
1.8 高速領域クリア
2.0　データルーティング
2.1 入力データフォーマッタ
2.1.1　ピクセル圧縮
2.1.2　入力データフォーマット



2.1.3　アキュムレータ
2.1.4　最終フォーマッティング
2.2 出力データフォーマッタ
2.2.1　RDAT、RPIX（８ビット、１６ビット、３２ビットピクセル)オペレーション
2.2.2　RPIX（６４ビットピクセル)オペレーション
2.2.3　RPIX（９６ビットピクセル)オペレーション
2.2.4　RPIX（１２８ビットピクセル)オペレーション
2.3 SRAMからピクセルALUへのルーティング
2.3.1　８ビット、１６ビットおよび３２ビットピクセルのSRAM編成
2.3.2　６４ビットピクセルのSRAM編成
2.3.3　９６ビットピクセルのSRAM編成
2.3.4　１２８ビットピクセルのSRAM編成
2.3.5　UnpackColors
2.3.6　UnpackDepths
2.3.7　UnpackExtras
2.3.8　UnpackAlpha, UnpackRed, UnpackGreen, UnpackBlue
2.3.9　UnpackDepth
2.3.10　UnpackStencil
2.3.11　UnpackWid
2.3.12　SramToPaluData
2.4 ピクセルALUからSRAMへのデータルーティング
2.4.1　PackColor
2.4.2　PackDepth
2.4.3　PackExtra
2.4.4　PaluToSramData
2.5 ピクセルALUからSRAMへのマスク生成
2.5.1　WriteEnableMask
2.5.2　ピクセルアドレスマスク
2.5.3　MaskDepth
2.5.4　EnableMask
2.5.5　SelectPlaneMask
2.5.6　ピクセルALUからSRAMへのマスク
3.0　ピクセルフォーマット
3.1 ８ビットピクセルフォーマット
3.2 １６ビットピクセルフォーマット
3.3 ３２ビットピクセルフォーマット
3.4 ６４ビットピクセルフォーマット
3.5 ９６ビットピクセルフォーマット
3.6 １２８ビットピクセルフォーマット
3.7 マルチ・サンプル・ポリゴン・アンチエイリアシング
3.7.1　累算バッファ
3.7.2　Aバッファ
3.7.3　マルチサンプル
3.7.3.1　サンプルあたり色のみ
3.7.3.2　サンプルあたり色およびデプス

3.7.4　　サンプルあたり色およびデプスの速度改良
3.8 ２５６ビットピクセルフォーマット（４×マルチサンプル）
3.9 ５１２ビットピクセルフォーマット（６×マルチサンプル）
4.0　双方向I/O
1.0 アーキテクチャ
図１は、計算機システムにおいて動作するグラフィックスサブシステム１００を図示する。グラフィックスサブシステム１００は、ビデオディスプレイフレームバッファとも呼ばれる。グラフィックスサブシステム１００は、レンダリングコントローラ１０２、その中でグラフィックスサブシステム１００が動作する計算機システムへのインタフェース１０４、ビデオ出力回路１０６、レンダリングコントローラ１０２からビデオ出力回路１０６へ延びるビデオ出力チャネル１０８、ここに開示する１つまたは２つ以上のデュアルピクセル３DRAMチップ１１０、１つまたは２つ以上のレンダリングバス１１２、および１つまたは２つ以上のアドレスおよび制御バス１１４を含む。図１に示すように、ビデオ出力回路１０６はレンダリングコントローラ１０２から物理的に分離されているが、代替的実施例は、単一のチップまたは装置の中にレンダリングコントローラ１０２およびビデオ出力回路１０６の両方を含む。
【００４０】図１に示すグラフィックスサブシステム１００は、レンダリングバス１１２とアドレスおよび制御バス１１４との２つの別個の対に接続される４つのデュアルピクセル３DRAMチップの組１１０ａ～dおよび１１０ｅ～ｈの２組を含む。ここでの構成では、レンダリングバス１１２とアドレスおよび制御バス１１４との対に接続できるデュアルピクセル３DRAMチップ１１０の最大数は４つである。しかし、図１の省略記号（...）によって示されるように、グラフィックスサブシステムにおいて使用されるレンダリングコントローラ１０２とデュアルピクセル３DRAMチップ１１０との間のレンダリングバスと制御バスとの対の数に制限はない。グラフィックスサブシステム１００は、処理されるピクセルデータに依存して、シングル・バッファまたはダブル・バッファのいずれかであり、色バッファＡおよびＢならびに単一のＺバッファを含む。グラフィックスサブシステム１００は、８ビット/ピクセルから最大５１２ビット/ピクセルまでの範囲のさまざまな異なったピクセルフォーマットをサポートする能力を特徴とし、これによって、多数のフレームバッファ１００の寸法に対応する。
【００４１】レンダリングコントローラ１０２は、アドレスおよび制御バス１１４を通じてデュアルピクセル３DRAMチップ１１０ａ～ｈのための制御情報を転送する。レンダリングコントローラ１０２は、レンダリングバス１１２を通じてデュアルピクセル３DRAMチップ１１０へのおよびデュアルピクセル３DRAMチップ１１０からのピクセルデータアクセスを行なう。レンダリングコントローラ１０２は、ピクセルアクセスのシーケンスをレンダリングオペレーションのシーケンスへ変換する。レンダリングバス１１２は、近年グラフィックス処理分野に出現したRDRAM™ およびSLDRAMなどの高帯域通信アーキテクチャをサポートする。



【００４２】レンダリングコントローラ１０２は、レンダリングバス１１２を通じてピクセルデータをデュアルピクセル３DRAMチップ１１０ａ～ｈに書込み、デュアルピクセル３DRAMチップ１１０ａ～ｈは、レンダリングバス１１２を通じてレンダリングコントローラ１０２へ更新されたピクセルデータを転送する。レンダリングコントローラ１０２は、別個のアドレスおよび制御バス１１４を通じてフレームバッファ制御信号およびフレームバッファコマンドをデュアルピクセル３DRAMチップ１１０ａ～ｈへ転送する。フレームバッファコマンドおよびフレームバッファ制御信号は、デュアルピクセル３DRAMチップ１１０ａ～ｈの内部オペレーションを調整する。
【００４３】デュアルピクセル３DRAMチップ１１０は、ピンの数を最小限に留めつつ、レンダリングコントローラ１０２とデュアルピクセル３DRAMチップ１１０ａ～ｈとの間で、制御情報を受信し、グラフィックスデータを送受信するため、高帯域入/出力（I/O）技術をサポートする。一実施例については、レンダリングバス１０２は、データおよび制御I/O用のRambus Direct RDRAM™ 仕様に準拠し、１．８ギガビット/秒でピクセルデータを転送する１８ビット双方向データバスと８００メガビット/秒で情報を転送する単方向アドレスおよび制御バスとを有する。他実施例については、レンダリングバス１０２は、公式にはSyncLinkとして知られるオープンIEEEおよびJEDEC規格、SLDRAMに準拠する。以下に挙げる刊行物は、このような高帯域I/Oアーキテクチャを詳細に説明しており、その全体としてここに引用により援用される。援用される文献は、ピーター・ギリンガム（PeterGillingham）による『SLDRAMのアーキテクチャおよび機能の概要』（“SLDRAM Architectural and Functional Overview”）、MOSAIDテクノロジーズ（MOSAID Technologies, Inc.）、１９９７年８月２９日；IEEEコンピュータ学会マイクロプロセッサおよびマイクロコンピュータ規格小委員会後援の『高速メモリインタフェース（SyncLink）規格草案』（“Draft Standard For A High-Speed MemoryInterface (SyncLink)”）、草案（Draft）0.99 IEEE p1596.7－199X、１９９６年；『４００Ｍｂ/ｓ/ピンSLDRAM″ ４M×１８SLDRAMパイプライン方式８バンク２．５Ｖ動作』（“400Mb/s/pin SLDRAM″ 4M×18 SLDRAM pipelined, eightbank, 2.5V operatI/On”）、Draft/Advance SLD4M18DR400 4 MEG×18 SLDRAM、SLDRAM協会発行、１９９７年９月２２日である。
もちろん、異なったI/Oアーキテクチャをサポートするためデュアルピクセル３DRAMチップ１１０の他実施例も

可能である。

【００４４】図２は、一実施例のデュアルピクセル３DRAMチップ１１０を示す。デュアルピクセル３DRAMチップ１１０は、図１のデュアルピクセル３DRAMチップ１１０ａ～ｈの各々と実質的に同様である。デュアルピクセル３DRAMチップ１１０は、DRAMアレイ１１６、SRAMピクセルバッファ１１８、２つのピクセル算術論理演算装置（ALU）１２０および１２１、ならびに図示される実施例においては別個のグローバル書込バス１２３およびグローバル読出バス１２５それぞれを含むグローバルバス１２２を含む。デュアルピクセル３DRAMチップ１１０はまた、グラフィックスデータが処理されデュアルピクセル３DRAMチップ１１０内にストアされるのに伴い、グラフィックスデータを割振りフォーマット化するデータフォーマッタといくつかのデータバスとを含む。

【００４５】グラフィックスデータは、レンダリングバス１１２を通じてデュアルピクセル３DRAMチップ１１０へ入りかつそこから出る。デュアルピクセル３DRAMチップ１１０へ送信されるグラフィックスデータは、入/出力バス（I/Oバス）１２４により受信される。入力グラフィックスデータは、入力データデマルチプレクサ１２６によりデマルチプレクスされ、入力データバス１２８を通じて入力データフォーマッタ１３０へ転送され、そこでデータはフォーマット化され、次にフォーマット化されたデータは、それぞれ処理のためにピクセルALU１２０および１２１へと入力データバス１２９および１３１を通じて送信される。

【００４６】デュアルピクセル３DRAMチップ１１０からレンダリングバス１１２へ送信されるグラフィックスデータは、SRAM出力データバス１３２を通じてSRAMピクセルバッファ１１８から送られる。レンダリングバス１１２上で受信されるまでに、グラフィックスデータは出力データフォーマッタ１３４によりフォーマット化され、出力データバス１３５を通じて送信され、出力データマルチプレクサ１３６によりマルチプレクスされ、チップ１１０からI/Oバス１２４を通じて転送される。

【００４７】データは、２つの別個のデータバス１３８および１３９を通じて、ピクセルALU１２０および１２１とSRAMピクセルバッファ１１８との間で割振られる。ピクセルALU１２０および１２１からSRAMピクセルバッファ１１８へと送信されるグラフィックスデータは、その途上、ピクセルALUからSRAMへのフォーマッタ１４０によりフォーマット化され、次にデータバス１４１を通じてSRAMピクセルバッファ１１８へと割振られる。

【００４８】SRAMピクセルバッファ１１８からピクセルALU１２０および１２１へと送信されるグラフィックスデータは、SRAM出力データバス１３２を通じてSRAMからピクセルALUへのフォーマッタ１４４へ割振られ、そしてフォーマット化されたデータはSRAMピクセルバッファデータバス１４２および１４３を通じてピクセルALU１２０および１２１へ送信される。

【００４９】チップ１１０上で行なわれるオペレーションを方向づけるために使用されるアドレスおよび制御情報は、アドレスおよび制御バス１１４を通じてデュアルピクセル３DRAMチップ１１０へ送信される。情報は、アドレスおよび制御入力バス１４６で受信され、アドレスおよび制御デマルチプレクサ１４８によりデマルチプレクスされ、アドレスおよび制御バス１５０に沿ってデコーダ１５１へと送信される。デコーダ１５１は、デマルチプレクスされたアドレスおよび制御情報を受信し、これをデコードし、次にデコードされた情報はピクセルALUオペレーションチャネル１５２を通じてピクセルALU１２０および１２１ならびにSRAMピクセルバッファ１１８へ送信され、グローバルバスオペレーションチャネル１５４を通じて、SRAMピクセルバッファ１１８、パイプラインレジスタ１２７および１３７ならびにDRAMアレイ１１６へと送信され、そして、バンクオペレーションチャネル１５６を通じてDRAMアレイ１１６へ送信される。

【００５０】図３は、RAMBUS™ または（以前にはSyncLinkとして知られていた）SLDRAM入力/出力インタフェース仕様のいずれかで動作するよう構成されるデュアルピクセル３DRAMチップ１１０の一実施例のダイサイズのフロアプランを示す。図２の機能ブロック図とは異なり、図３のデュアルピクセル３DRAMチップ１１０のレイアウトは、図２に示す機能的要素のいくつかがデュアルピクセル３DRAMチップ１１０の特定の実施例においてどのように物理的に実現され得るのかを示す。

【００５１】たとえば、デュアルピクセル３DRAMチップ１１０のこの物理的実現例は、チップ１１０の４つの角に位置する４つのセクション１１６ａ～ｄへ物理的に分離されたDRAMアレイ１１６を特徴とする。この物理的な分離にもかかわらず、DRAMアレイ１１６は図２に示すように１つの機能的単位として動作する。図３のDRAMアレイ１１６は、図３の４つのセクション１１６ａ～ｄの各々の中のＡ～Ｈとラベリングされる８つのインターリーブされたモジュラーDRAMバンク１５８を含む。DRAMアレイ１１６と同様、８つのDRAMバンク１５８Ａ～Ｈはチップ１１０の４つの角に物理的に配置されるが、これらは、（３２個ではなく）８個の機能単位として動作する。

【００５２】デュアルピクセル３DRAMチップ１１０の全体としてのDRAM容量は、特定のチップ１１０の構成において使用されるモジュラーDRAMバンク１５８の数に依存して幅がある。各DRAMバンク１５８は、センス増幅器１６０を含む１組のラインバッファを含む。図２２および図２３を参照されたい。（「ラインバッファ」および「センス増幅器」または「センスアンプ」という語は、ここでは交換可能なものとして使用され、いずれも要素１６０を指す。）各DRAMバンク１５８は、DRAMビットのいくつかのライン１６４を含む複数のDRAMページ１６２

を含む。

【００５３】デュアルピクセル３DRAMチップ１１０の一実施例は、１０個のインターリーブされたDRAMバンク１５８を含む全部で４０メガビットのDRAMアレイ１１６を有し、各バンク１５８は５１２個のページ１６２を含み、各ページ１６２は８個の１０２４ビットライン１６４を含む。この実施例においては、各DRAMバンク１５８の構造は一定して４メガビットDRAMである（１０２４ビット/ライン*８ライン/ページ*５１２ページ/バンク＝４１９４３０４ビット/バンク≒４メガビット/バンク）。デュアルピクセル３DRAMチップ１１０内のインターリーブされたDRAMバンク１５８の数を変えることによって、チップの内部アーキテクチャを変更することなくチップ１１０の総記憶容量を調整することができる。

【００５４】ラインバッファ１６０はセンス増幅器を含み、DRAMバンク１５８内にストアされたピクセルデータにアクセスするとき、キャッシュラインの第２のレベルとして働く。（キャッシュラインの第１のレベルは、SRAMピクセルバッファ１１８内のメモリのラインである。）ラインバッファ１６０はDRAMバンク１５８へと直接マッピングされる。一実施例においては、各ラインバッファ１６０が、対応するDRAMバンク１５８のページの１つをマッピングする。一実施例においては、ラインバッファのエントリは１０２４ビットのライン一つを含む。

【００５５】再び図２を参照し、ピクセルバッファ１１８は高速マルチポートスタティックRAM（SRAM）構成要素である。データは、グローバルバス１２２を通じて、SRAMピクセルバッファ１１８とDRAMアレイ１１６との間を転送される。図示される実施例においては、グローバルバス１２２は２つの単方向バス、グローバル書込バス１２３とグローバル読出バス１２５とを含む。

【００５６】SRAMピクセルバッファ１１８は、ピクセルALUデータバス１３８を通じてピクセルALUからSRAMへのフォーマッタ１４０によりフォーマット化されたデータを、読出す。SRAMピクセルバッファ１１８は、SRAM出力データバス１３２を通じて、出力データフォーマッタ１３４およびSRAMからピクセルALUへのフォーマッタ１４４の両方にデータを書込む。出力データフォーマッタ１３４は、SRAMピクセルバッファ１１８からデータフィールドをアンパックし、レンダリングバス１１２を通じて送信されるディスプレイ出力用にフィールドのいくつかを再パックする。SRAMからピクセルALUへのフォーマッタ１４４はまた、ピクセルALU１２０および１２１が使用するようデータフィールドをアンパックする。

【００５７】一実施例においては、グローバル書込バス１２３およびグローバル読出バス１２５は各々、SRAMピクセルバッファ１１８とDRAMアレイ１１６との間で１０２４ビットを搬送し、一方、ピクセルALUデータバス１３８および１３９ならびにSRAM出力データバス１３２は各々２５６ビット幅である。

【００５８】一実施例においては、SRAMピクセルバッファ１１８は、８本のキャッシュラインを有し、各キャッシュラインはメモリ１０２４ビット（１キロビット）を含む。８キロビットSRAMピクセルバッファ１１８は８つの１キロビットキャッシュラインに編成される。他実施例においては、SRAMピクセルバッファ１１８は１６本の１キロビットキャッシュラインに編成される。

【００５９】グローバルバス１２２は、SRAMピクセルバッファ１１８とDRAMアレイ１１６のセンスアンプ１６０との間での通信を可能にする。好ましい実施例においては、グローバルバス１２２は１０２４ビット１０ナノ秒デュアルバス１２３および１２５を含む。グローバル読出バス１２５は、読出パイプラインレジスタ１２７を通じてセンスアンプ１６０からSRAMピクセルバッファ１１８へデータを転送し、グローバル書込バス１２３は、データ書込パイプラインレジスタ１３７を通じてSRAMピクセルバッファ１１８からセンスアンプ１６０へピクセルデータおよびマスクデータを転送する。この実施例はまた、DRAMアレイ１１６内のどのビットに上書きするかを制御するため、マスク書込パイプラインレジスタ１４５およびマスク書込バス１４７を用いる。他実施例においては、グローバルバス１２２は、グローバル読出バス１２５およびグローバル書込バス１２３の両方を含むが、パイプラインレジスタ１２７、１３７および１４５は用いられない。さらに他の実施例においては、グローバルバス１２２は、SRAMピクセルバッファ１１８からの読出およびSRAMピクセルバッファ１１８への書込の両方のために交互に使用される単一の双方向バスを含む。

【００６０】ピクセルALU１２０および１２１とSRAMピクセルバッファ１１８との間のデータ転送は、グローバルバス１２２を通じてのSRAMピクセルバッファ１１８とDRAMアレイ１１６との間のデータ転送とは異なる。一実施例においては、ピクセルALU１２０および１２１は、２５６ビット５ナノ秒のデータバス１３８および１３９を通じてデータを書込み、ピクセルALU１２０および１２１は２５６ビット５ナノ秒のバス１４２および１４３を通じて送信されるデータを読出す。

【００６１】1.1 ピクセルALU

ピクセルALU１２０および１２１は、SRAMピクセルバッファ１１８へのパイプライン方式でのリード・モディファイ・ライトオペレーションを可能にする。パイプライン方式のリード・モディファイ・ライトオペレーションは、Ｚ－バッファ比較、RGBアルファラスタオペレーション、およびブレンドオペレーションを含む。好ましい実施例のSRAMピクセルバッファ１１８のマルチポート性により、グローバルバス１２２を通じてのDRAMアレイ１１６のラインバッファ１６０とSRAMピクセルバッファ１１８との間での全キャッシュラインの並列転送が可能になる。

【００６２】図２および図３に図示するように、デュアルピクセル３DRAMチップ１１０は、オフチップ帯域幅要求を最小限にするための２つのオンチップピクセルALU１２０および１２１を特徴とする。９６ビットまたは１２８ビットのピクセルを更新するためには、４０ビットの色情報と３２ビットのデプス情報の送信が必要である。

【００６３】デュアルピクセル３DRAMグラフィックスサブシステム１００は、チップ１１０上のピンの数を最小限にしつつ、レンダリングコントローラ１０２とデュアルピクセル３DRAMチップ１１０との間でデータおよび制御情報を送信するため、Direct RDRAM™ BY RAMBUSまたは（従前にはSyncLinkとして知られていた）SLDRAMなどの高帯域I/O技術を用いる。一実施例においては、デュアルピクセル３DRAMグラフィックスサブシステム１００は、１．８ギガビット/秒で遷移する（すなわち立上がり端および立下がり端の両方におけるデータの遷移が４００MHzクロックである）１８ビット半二重双方向データバス１１２、および、８００メガビット/秒で遷移するレンダリングコントローラ１０２からデュアルピクセル３DRAMチップ１１０ａ～ｈへの８ビット単方向制御バス１１４を使用する。この実施例においては、ピクセルALU１２０および１２１は、処理されるピクセルのフォーマットに依存して、２００MHzまたは１３３MHzのいずれかで動作し、そのため、デュアルピクセル３DRAMチップ１１０のピンにおいて受信される狭高周波数データストリームは、内部では４倍から６倍幅が広いデータストリームへとデマルチプレクスされる必要がある。同様に、デュアルピクセル３DRAMチップ１１０において処理されるデータは、レンダリングバス１１２を通じてレンダリングコントローラ１０２へ送られる前にマルチプレクスされねばならない。

【００６４】図３を参照し、ピクセルALU１２０および１２１は、デュアルピクセル３DRAMチップ１１０の中央に位置づけられる。ピクセルALU１２０および１２１は、処理要素の２つの完全な組を含み、したがって、多くの状況下でピクセルALU１２０および１２１がオペレーションごとに２つのピクセルを処理することを可能にする。処理されるピクセルのフォーマットが、ピクセルALU１２０および１２１が２つのピクセルを個別に処理できるか、または、一度に単一のピクセルを処理するようそれらのリソースを組合せなければならないかどうかを決定する。処理要素の完全な組ひとつは、４つのラスタオペレーション（ROP）/ブレンドユニット１６６、デプスユニット１６８、ステンシルユニット１７０およびウィンドウ識別（WID）ユニット１７２を含む。

【００６５】図４は、処理要素を２組含むピクセルALU１２０および１２１の一実施例を示す。処理要素の第１の組は、第１のピクセルALU１２０に対応し、「０」とラベリングされている。処理要素の第２の組は、第２のピクセルALU１２１に対応し、「１」とラベリングされている。ROP/ブレンドユニット１６６は、ユニット１６６を実現するため使用される回路のいくつかが共用されることを示すため図では重なり合うように示されている。

【００６６】図５はやはり処理要素の完全な組２つを含むピクセルALU１２０および１２１の他実施例である。ROP/ブレンドユニット１６６は、別個のROPユニット１７４とブレンドユニット１７６とに分割される。この実施例では、ROP/ブレンドユニット１６６の間で共有される回路はない。この実施例は、４つの８ビットブレンドユニット１７８と４つの１０ビットブレンドユニット１８０とを特徴とする。より大きなピクセルフォーマットに対応するため、容量の異なるブレンドユニットが設けられる。好ましい実施例では、ROP/ブレンドユニットは各々１０ビットユニットである。

【００６７】図６は、ピクセルALU１２０および１２１のいずれか一方の中の処理ユニットの完全な組を示す。チップ１１０上に組合される、ピクセルALU１２０および１２１はこれらユニットの完全な組２つを有し、したがって、デュアルピクセル３DRAMチップ１１０が多くの状況下でオペレーションごとに２つのピクセルを処理することができる。文字「S」は、レンダリングバス１１２を通じてピクセルALU１２０または１２１へ送信されるソースデータを示す。文字「D」は、グローバルバス１２２を通じてSRAMピクセルバッファ１１８からピクセルALU１２０または１２１へ送信される行先データを示す。文字「R」は、グローバルバス１２２を通じてSRAMピクセルバッファ１１８へ再び送信される結果データを示す。文字「DT」、「ST」および「WT」は、それぞれデプスユニット１６８、ステンシルユニット１７０およびウインドウIDユニット１７２により行なわれるテストの結果である。処理ユニットの完全な組は、各ピクセルのアルファ成分、赤成分、緑成分および青成分の処理のための４つのROP/ブレンドユニット１６６を含む。デュアルピクセル３DRAMチップ１１０のこの実施例においては、ROP/ブレンドユニット１６６へのソースデータ入力ストリームおよび行先データ入力ストリームは各々１１ビット幅である。ROP/ブレンドユニット１６６から出力される結果データストリームは、１０ビット幅である。デプスユニット１６８については、ソースデータストリーム、行先データストリーム、および結果データストリームは３２ビット幅である。ステンシルユニット１７０は、行先ストリームを受信して、結果ストリームを出力し、その両方とも８ビット幅である。ウインドウIDユニット１７２は８ビットの行先データストリームを受信する。

【００６８】1.1.1　ROP/ブレンドユニット

図７は、一実施例における１つのROP/ブレンドユニット１６６をブロック図の形で示す。この実施例において

は、８個の１０ビットROP/ブレンドユニット１６６が２つのピクセルALU１２０および１２１の中に位置する。８個のROP/ブレンドユニット１６６の各々は、機能的に同一であり、（ソース色（Sc）、ソース係数（Sf）、行先色（Dc）および行先係数（Df）用の）４つの１１ビットデータ入力と結果用の１０ビットデータ出力とを有する。各ROP/ブレンドユニット１６６は、（１）ROP（Sc、Pc、Dc）、（２）min（Sc、Dc）、（３）max（Sc、Dc）、（４）Sc＊Sf＋Dc＊Df、（５）Sc＊Sf－Dc＊Df、または（６）Dc＊Df－Sc＊Sfの６つのオペレーションのうち１つを行なう。

【００６９】ROPオペレーション、minオペレーション、maxオペレーションまたは８ビットブレンドオペレーションを行なうとき、８個のROP/ブレンドユニット１６６＊は、すべて並列に作業できる。１０ビットブレンドオペレーションを行なうときは、必要とされる処理を行なうためにブレンドユニット１７６の対を互いに組合せる必要がある。したがってデュアルピクセル３DRAMチップ１１０により１０ビットブレンドオペレーションが行なわれるときには、１度に１つのピクセルしか処理できない。他実施例は、８個の１０ビットROP/ブレンドユニット１６６を有し、いかなる場合にも一度に２つのピクセルを処理できる。

【００７０】ソースブレンド係数「Sf」および行先ブレンド係数「Df」は、次の表１に示すようにソース色、行先色およびパターン色から導出される。

【００７１】

【表１】

| パラメータ | 係数 | | | | Sf | Df |
|---|---|---|---|---|---|---|
| | アルファ | 赤 | 緑 | 青 | | |
| GL_ZERO | 0 | | | | x | x |
| GL_ONE | 1 | | | | x | x |
| GL_SRC_COLOR | Sα | SR | SG | SB | | x |
| GL_ONE_MINUS_SRC_COLOR | 1-Sα | 1-Sr | 1-Sg | 1-Sb | | x |
| GL_DST_COLOR | Dα | Dr | Dg | Db | x | |
| GL_ONE_MINUS_DST_COLOR | 1-Dα | 1-Dr | 1-Dg | 1-Db | x | |
| GL_SRC_ALPHA | Sα | | | | x | x |
| GL_ONE_MINUS_SRC_ALPHA | 1-Sα | | | | x | x |
| GL_DST_ALPHA | Dα | | | | x | x |
| GL_ONE_MINUS_DST_ALPHA | 1-Dα | | | | x | x |
| GL_SRC_ALPHA_SATURATE | 1 | min(Sα, 1-Dα) | | | x | |
| CL_CONSTANT_COLOR_EXT | Pα | Pr | Pg | Pb | x | x |
| CL_ONE_MINUS_CONSTANT_COLOR_EXT | 1-Pα | 1-Pr | 1-Pg | 1-Pb | x | x |
| CL_CONSTANT_ALPHA_EXT | Pα | | | | x | x |
| CL_ONE_MINUS_CONSTANT_ALPHA_EXT | 1-Pα | | | | x | x |

【００７２】次に図８を参照し、ROP/ブレンドユニット１６６のラスタオペレーション（ROP）部１７４が、２５６個のブール演算のうち１つをソースSc入力、行先Dc入力およびパターンPc入力に対して行なう。アドレスおよび制御バス１１４を通じて送信される情報によりセットされるROPレジスタ１８２が、２５６個のブール演算のうちどれを行なうかを決定する。３つの入力が必要と※される場合には、ラスタオペレーションが行なわれる前に入力ひとつがパターンレジスタ１８４に書込まれる。ROP部１７４の１ビットスライスは、次のブール方程式によって実現できる。

【００７３】

【数１】

```
Result = (Op[0] & ~Dc & ~Sc & ~Pc) | (Op[1] & Dc & ~Sc &
~Pc) |
(Op[2] & ~Dc & ~Sc & ~Pc) | (Op[3] & Dc & ~Sc & ~Pc) |
(Op[4] & ~Dc & ~Sc & ~Pc) | (Op[5] & Dc & ~Sc & ~Pc) |
(Op[6] & ~Dc & ~Sc & ~Pc) | (Op[7] & Dc & ~Sc & ~Pc) ;
```

【００７４】図９から図１４を参照し、ROP/ブレンドユニット１６６のブレンド部１７６は、処理回路または要素すなわち、１つのディザ計算装置１８６、２つの乗算器１８８および１９０、加算器１９２、１つの切捨て装置１９４および１つのクランプ装置１９６を含む。図９に、８ビットブレンドユニット１７８を示す。オペレー

ションごとに２つのピクセル（またはアンチエイリアシングを行なうときにはサンプル）をブレンドできるようにするためにはこれらのユニットが８つ必要である。

【００７５】ソース色値S colorおよび行先色値D colorは、それらを表わすためにいくつのビットが使用されているかにかかわらず、[0.0,1.0]の範囲内の値をとる。各ビットエンコーディングはある範囲の値を表わす。たとえば、８ビット入力で１４は、[14/256,15/256）の範囲を表わす。計算を行なうときは、範囲全体を表わすための一つの値が選択される。もし範囲[14/256,15/256）を表わすために値14/256が選択されたならば、計算のエラーはその範囲の下端に向けて偏らされるであろう。もし、その範囲の中間点が選択されたならば、すなわち14.5/256が選択されたならば、計算のエラーはその範囲に対して偏りがないようにされ、最終的な結果はより正確になるであろう。これは乗算器への入力の最下位ビットに１を連結することによって達成できる。したがって８ビットブレンドユニット１７８は、９ビット対９ビットの乗算器１８８および１９０を必要とする。

【００７６】図１０は、ブレンドオペレーション計算の間の中間値のフォーマットを示す。ディザ・オフセット値が、ピクセルのXアドレスおよびYアドレスの２つの最下位ビットに基づいて計算される。２つの積とディザオフセットとが加算される。最大で、積の値の一方の否定がとられてもよい。次に和が切捨てられ、クランプされて結果となる。

【００７７】一実施例においては、図１１およびここにその全体として引用により援用されるフォーリー（Foley）、ヴァンダム（vanDam）、フェイナー（Feiner）、およびヒュー（Hughes）による『コンピュータグラフィックスの原理および実際』（“Computer Graphics Principles and Practice”）、第2版ｐｐ．５７０～５７１に反映されるように、４×４Bayerディザマトリクスのドット分散型組織的ディザアルゴリズムが用いられる。(-0.5,0.5）の範囲のディザ値が切捨て前に結果値をオフセットする。もちろん、当分野で公知の他のディザアルゴリズムも使用できる。

【００７８】図１２は１０ビットブレンドユニット１８０を示す。オペレーションごとに1つのピクセルをブレンドできるようにするためこれらのユニット4つが必要である。一実施例においては、１０ビットブレンドユニット１８０各々と８ビットブレンドユニット１７８の各対との間で可能な限り多くの論理を再使用するような態様で、１０ビットブレンドユニット１８０が実現される。１０ビットブレンドユニット１８０は、８ビットブレンドユニット１７８に関して上に説明したのと同じ理由で２つの１１ビット対１１ビット乗算器１８８および１９０を必要とする。

【００７９】この実施例はまた、そこを通ってソース係数Sfおよび行先係数Dfのデータが ROP/ブレンドユニット１６６へ入力される２つの入力マルチプレクサ２０６および２０８を特徴とする。図１４は、入力マルチプレクサ２０６および２０８の詳細な図である。

【００８０】1.1.2 デプスユニット

図１５はデプスユニット１６８のブロック図である。ピクセルALU１２０および１２１には２つのデプスユニット１６８がある。２つのデプスユニット１６８は、ソースデータ３２ビットを行先データ３２ビットと比較する。制御情報が１６ビットマスクレジスタ２１０へ与えられ、次に、比較オペレーションの前にソースデータおよび行先データとビットごとに論理積を取られる。(1)フェール、(2)src＜dest、(3)src＝＝dest、(4)src＜＝dest、(5)src＞dest、(6)src！＝dest、(7)src＞＝dest、および(8)パスの８つのテストのうち１つを指定する、ファンクション/オペレーションレジスタ２１２内の３ビットレジスタフィールドにより、符号なし整数比較オペレーションが指定される。正のIEEE単精度浮動小数点数であれば、符号ビットをゼロでマスクすれば、正しく比較されるであろう。

【００８１】1.1.3 ステンシルユニット

図１６および図１７は、ステンシルユニット１７０のブロック図である。ピクセルALU１２０および１２１には２つのステンシルユニット１７０がある。２つの８ビットステンシルユニット１７０は各々、行先ステンシル用の８ビットデータ入力、１ビットデプステスト入力、８ビットデータ出力および１ビット比較出力を有する。符号のない整数比較オペレーションは、ファンクション/オペレーションレジスタ２１４内の３ビットレジスタフィールドにより指定され、(1)フェール、(2)ref＜dest、(3)ref＝＝dest、(4)ref＜＝dest、(5)ref＞dest、(6)ref！＝dest、(7)ref＞＝dest、および(8)パスの８つのテストのうち１つを指定する。

【００８２】図１７を参照し、デプステストおよびステンシルテストの状態に依存して、３つのステンシルオペレーションコードのうち１つが選択される。オペレーションコードは、どのステンシルオペレーションが行なわれるかを決定する。可能なステンシルオペレーションは、dest、0、ref、wrap(dest＋1)、wrap(dest－1)、saturate(dest＋1)、saturate(dest－1)、および~destである。

【００８３】1.1.4 ウインドウIDユニット

図１８は、ウインドウ識別（ID）ユニット１７２のブロック図である。ピクセルALU１２０および１２１には２つのウインドウIDユニット１７２がある。２つの８ビットウインドウID比較ユニット１７２は各々、行先WID用の８ビットデータ入力と１ビット比較結果出力とを有する。ウインドウIDユニット１７２の挙動は、マスクレジスタ２１６内の８ビットフィールドと基準レジスタ２１８内の８ビットフィールドとファンクションレジスタ２２０内の３ビットフィールドとによって制御される。フ

ァンクションレジスタの２ビットフィールドは、(1)フェール、(2)ref＜dest、(3)ref＝＝dest、(4)ref＜＝dest、(5)ref＞dest、(6)ref！＝dest、(7)ref＞＝dest、および(8)パスの８つのテストのうち１つを指定する。ウインドウIDユニット１７２とステンシルテストユニット１７０とは機能的に同一である。

【００８４】ウインドウIDユニット１７２、ステンシル＊ユニット１７０およびデプスユニット１６８からのテスト結果は、デュアルピクセル３DRAMチップ１１０内のオペレーションを制御するため使用される。表２は、３つのユニットの結果に基づいて行なわれるかまたは行なわれないアクションのリストである。

【００８５】

【表２】

表２　ピクセルテスト

| WID テスト | ステンシルテスト | デプステスト | アクション |
|---|---|---|---|
| フェール | -- | -- | ピクセルを書込まない |
| パス | フェール | -- | ステンシルビットのみを書込む |
| パス | パス | フェール | ステンシルビットのみを書込む |
| パス | パス | パス | ステンシルビット、デプスビット、および色ビットを書込む |

【００８６】1.2　SRAMピクセルバッファ図１９および図２０は、SRAMピクセルバッファ１１８を示す。一実施例においては、SRAMピクセルバッファ１１８は、８ワード×１０２４ビットのマルチポートSRAMで実現される。グローバルバス１２２は、１０２４ビットパイプラインレジスタ１２７を通じてDRAMアレイ１１６のセンスアンプ１６０からSRAMピクセルバッファ１１８へデータを転送する１０２４ビット１０ナノ秒読出バス１２５を含む。グローバルバス１２２はまた、１０２４ビットパイプラインレジスタ１３７を通じてSRAMピクセルバッファ１１８からセンスアンプ１６０へデータを転送する１０２４ビット１０ナノ秒書込バス１２３を含む。この実施例においては、グローバルバス１２２はまた、マスク書込パイプラインレジスタ１４５およびマスク書込バス１４７を通じてSRAMピクセルバッファ１１８からセンスアンプ１６０へマスクデータ１０２４ビットを転送する。

【００８７】SRAMピクセルバッファ１１８内のキャッシュラインは各々、バンクアドレス５ビットおよびコラムアドレス３ビットを含むタグ２３０と関連づけられる。タグ２３０は、現在SRAMピクセルバッファ１１８内にストアされているデータがやってきた位置を追跡するために使用される。

【００８８】グローバルバスの読出オペレーションは、Read Cache Line（RL：キャッシュライン読出）オペレーションまたはChange Cache Line（CL：キャッシュライン変更）オペレーションのいずれかにより開始される。初めの１０ナノ秒サイクルの間に、指定されたDRAMバンク１５８およびコラムからデータ読出パイプラインレジスタ１２７へとデータ１０２４ビットがコピーされる。次の１０ナノ秒サイクルの間に、データ読出パイプラインレジスタ１２７からSRAMピクセルバッファ１１８内の指定されたラインへとデータ１０２４ビットがコピーされ、そのデータが取出されたバンクおよびコラムのアドレスがキャッシュラインのタグ２３０へ書込まれる。

【００８９】グローバルバスの書込オペレーションは、Write Cache Line（WL：キャッシュライン書込）オペレーション、Masked Write Cache Line（ML：キャッシュラインのマスク書込）オペレーションまたはChange Cache Line（CL）オペレーションにより開始される。初めの10ナノ秒サイクルの間に、SRAMピクセルバッファ１１８内の指定されたラインからデータ書込パイプラインレジスタ１３７へとデータ１０２４ビットがコピーされ、プレーンマスクおよびバイトマスクレジスタからマスクデータ１０２４ビットが発生され、マスク書込パイプラインレジスタ１４７へコピーされる。もしオペレーションがWLまたはMLであれば、オペレーションのバンクおよびコラムアドレスは、バンクパイプラインレジスタ２３１およびコラムパイプラインレジスタ２３２へコピーされる。もしオペレーションがCLであれば、バンクパイプラインレジスタ２３１およびコラムパイプラインレジスタ２３２は、指定されたキャッシュラインのバンクおよびコラムタグ２３０からコピーする。次の１０ナノ秒サイクルの間に、データ書込パイプラインレジスタ１３７の内容は、マスク書込パイプラインレジスタ１４５の制御下で、バンクおよびコラムパイプラインレジスタ２３１および２３２により指定されるDRAMバンクおよびコラムへコピーされる。

【００９０】上に説明した手順に対するサンプル関数は以下のとおりである。

【００９１】

【数２】

```
function Mask[1023:0]
input [511:0] PlaneMask;
input [127:0] Bytemask;
input         IsMasked;
integer I;
begin
     for (i = 0; i < 1024; i = i+1) begin
          Mask[I]   =   (!IsMasked)   |   (PlaneMask[i%512]   &
     ByteMask[i/8]);
     end;
end;
endfunction
```

【００９２】ピクセルALU読出オペレーションは、Read Data（RDAT：データ読出）オペレーション、Read Pixel（RPIX：ピクセル読出）オペレーション、Single Pixel（SPIX：シングル・ピクセル）オペレーションおよびDual Pixel（DPIX：デュアル・ピクセル）オペレーションにより開始される。ピクセルALUデータバス１４１は、２５６ビット幅であり５ナノ秒の速度で動作し、SRAMピクセルバッファデータバス１４２もまた、２５６ビット幅であり５ナノ秒の速度で動作する。SRAMピクセルバッファ１１８は、５１２ビットのプレーンマスク２２２を含む付加的なラインを有する。MLオペレーションの間に、５１２ビットのプレーンマスクレジスタと１２８ビットのバイトマスクレジスタとが組合されて、１０２４ビットのマスク書込１４５の内容が発生され、これは同時に書込データとしてラッチされる。

【００９３】読出ポートおよび書込ポートは、１２８ビットのバウンダリでアドレス指定される。２５６ビットデータチャネル１３８および１４２は２００MHzで動作し、一方１０２４ビットグローバルバスチャネル１２２は１００MHzで動作する。好ましい実施例においては、SRAMピクセルバッファ１１８は、８つのキャッシュライン２２４を有する。２５６ビットデータチャネル１３８および１４２は、任意の１２８ビットバウンダリから始めて、キャッシュライン２２４の連続した２５６ビットにアクセスすることができねばならない。図３に示すように、一実施例においてこの要求は、SRAMピクセルバッファ１１８を、６４ビットのデータ読出チャネル１３８a～dおよびデータ書込チャネル１３２a～dを備える４つの物理アレイ１１８a～dに分けることによって満足される。この実施例においては、別個の読出アドレスおよび書込アドレスが、４つの別個のアレイ各々に与えられる。他実施例においては、グローバルバス１２２が、SRAMピクセルバッファ１１８とセンスアンプ１６０との間での同時読出および書込オペレーションを可能にし、それによってダーティタグの必要性をなくしている。

【００９４】他実施例においては、データがDRAMアレイ１１６から転送されてから、SRAMピクセルバッファキャッシュライン２２４のどのビットがピクセルALU１２０および１２１により更新されたかを示すため、SRAMピクセルバッファ内でダーティタグ２２６が使用される。図２０を参照し、ダーティタグSRAM２２６は、１６ワード×２５６ビットのデュアルポートSRAMで実現される。グローバルバス１２２は、２５６ビット１０ナノ秒の読出/書込ポートに接続される。ピクセルALU１２０および１２１は、２５６個のビットごとの書込イネーブル（WE）を備えるマスクデータを２５６ビット５ナノ秒書込ポートに書込む。

【００９５】Read Cache Line（RL）オペレーションの間に、データのラインは、SRAMピクセルバッファ１１８のデータ部に書込まれ、タグSRAM２２６内の対応するラインがクリアされる。

【００９６】1.3 メモリ構成

図２１、図２２および図２３は、一実施例のDRAMアレイ１１６の構成を示す。DRAMアレイ１１６はモジュラーDRAMバンク１５８を含む。メモリの基本単位は、１０２４ビットを保持するライン１６４である。ページ１６２は、８つのライン１６４、または８キロビットを含む。一実施例においては、DRAMバンク１５８は５１２個のページ１６２、またはメモリ４メガビットを含む。他実施例においては、DRAMバンク１５８は、１０２４個のページ１６２、またはメモリ８メガビットを含む。DRAMバンク１５８がメモリ８メガビットを保持する実施例においては、それぞれ４０メガビット、４８メガビット、６４メガビット、８０メガビット、９６メガビット、１２８メガビットおよび１６０メガビットのDRAMアレイ１１６をサポートするため５、６、８、１０、１２、１６および２０個の個別のDRAMバンク１５８が必要である。

【００９７】次の表は、64/72メガビットDirect RDRAM™のデータシートおよび4メガビット×18SLDRAMデータシートの両方に対するDRAMアレイ１１６の構成に基づく。Direct RDRAM™は、DRAMコアとI/Oセクションとの間の128/144ビットインタフェースを有する。SLDRAMは、64/72ビットインタフェースを有する。デュアルピクセル3DRAM１１０は、１０２４ビットインタフェースを有する。

【００９８】

【表３】

35 36

表３ DRAM構成（装置あたり）

| | 32 メガ ビット | 40 メガ ビット | 64 メガ ビット | 80 メガ ビット | 128 メガ ビット | 160 メガ ビット |
|---|---|---|---|---|---|---|
| DRAMあたりのバンク数 | 8 | 10 | 16 | 20 | 32 | 40 |
| バンクあたりのページ数 | 512 | 512 | 512 | 512 | 512 | 512 |
| ページあたりのライン数 | 8 | 8 | 8 | 8 | 8 | 8 |
| ラインあたりのビット数 | 1024 | 1024 | 1024 | 1024 | 1024 | 1024 |

【００９９】 ＊10＊【表４】

表４ DRAMバンクおよびページの可能な編成

| ビット/装置 | バンク/装置 | ページ /装置 | ビット /ページ | センスアンプ | 2レベル キャッシュ |
|---|---|---|---|---|---|
| 40 メガビット | 10doubled | 512 | 8K | 44K | 40K |
| | 5 | 1025 | 8K | 60K | 40K |
| | 10 | 512 | 8K | 80K | 80K |
| | 5 | 512 | 16K | 80K | 80K |
| 80 メガビット | 20doubled | 512 | 8K | 84K | 80K |
| | 5 | 1024 | 16K | 100K | 80K |
| | 10 | 1024 | 8K | 120K | 80K |
| | 20 | 512 | 8K | 160K | 160K |
| | 10 | 512 | 16K | 160K | 160K |
| | 5 | 512 | 32K | 160K | 160K |
| | 4 | 512 | 40K | 160K | 160K |
| 160 メガビット | 40doubled | 512 | 8K | 164K | 160K |
| | 5 | 1024 | 32K | 180K | 160K |
| | 10 | 1024 | 16K | 200K | 160K |
| | 20 | 1024 | 8K | 240K | 160K |
| | 40 | 512 | 8K | 320K | 320K |
| | 20 | 512 | 16K | 320K | 320K |
| | 10 | 512 | 32K | 320K | 320K |
| | 8 | 512 | 40K | 320K | 320K |
| | 5 | 512 | 64K | 320K | 320K |
| | 4 | 512 | 80K | 320K | 320K |

【０１００】1.4 ピン構成 ※DRAM™と互換性のあるピン構成を示す。

デュアルピクセル３DRAMチップ１１０は、Direct RDRAM 【０１０１】

™またはSLDRAMとピン互換性がある。次の表はDirect R※30 【表５】

表５ デュアルピクセル 3DRAM ピン

| 信号 | I/O | タイプ | 記 述 |
|---|---|---|---|
| RQ[7:0] | I | RSL | 制御およびアドレス情報 |
| DQA[8:0] | I/O | RSL | データ バイトA |
| DQB[8:0] | I/O | RSL | データ バイトB |
| CFM | I | RSL | マスタからのクロック+ |
| CFMN | I | RSL | マスタからのクロック- |
| CTM | I | RSL | マスタへのクロック+ |
| CTMN | I | RSL | マスタへのクロック- |
| $V_{REF}$ | | | RSL 信号に対する論理しきい値基準電圧 |
| $V_{TERM}$ | | | RSL ロードレジスタに対するターミネート電圧 |
| SIO[1:0] | I/O | CMOS | シリアル入力/出力 |
| CMD | I | CMOS | シリアルコマンド入力 |
| SCK | I | CMOS | シリアルクロック入力 |
| $V_{DD}$ | | | RDRAM コアおよびインタフェースに対する電源電圧 |
| GND | | | RDRAM コアおよびインタフェースに対する接地基準 |

【０１０２】1.5 プロトコル ™は８つのピンを備え、SLDRAMは１０個のピンを備え

制御およびアドレス情報を送信するため、Direct RDRAM 50 る。デュアルピクセル３DRAMプロトコルは、いずれのイ

ンタフェース技術においても使用できるよう８つのピンを必要とする。各ポートは、１つから４つのデュアルピクセル３DRAMチップ１１０を制御する。

【０１０３】次に図２４および図２５を参照し、制御およびアドレス情報８００メガバイト/秒が、３つのチャネルに分割される。ピクセルALUオペレーションチャネル１５２は、１秒あたり４００メガバイトを扱い、ピクセルALU１２０および１２１、SRAMピクセルバッファ１１８ならびにレンダリングバス１１２とのインタフェースを制御する。バンクオペレーションチャネル１５６は、１秒あたり２００メガバイトを扱い、DRAMバンク１５８およびページ１６２へのアクセスおよびプリチャージを制御する。グローバルバスオペレーションチャネル１５４は、１秒あたり２００メガバイトを扱い、グローバルバス１２２を通じてのSRAMピクセルバッファ１１８＊キャッシュライン224の読出および書込を制御する。各チャネル１５２、１５４および１５６は、３つのチャネルが同時に異なるデュアルピクセル３DRAMチップ110に対して動作できるよう、それ自身のオペレーションのフレーム化を行ないそれ自身の装置識別情報を含む。

【０１０４】1.5.1 DRAMバンクオペレーション
DRAMバンクオペレーションは、帯域幅２００メガバイト/秒の専用２ピン制御チャネル１５６を通じて送信される。DRAMバンクオペレーションは、任意のクロックの立下がり端において送信を開始できる。次の表６に挙げるバンクオペレーションが、一実施例において規定される。

【０１０５】
【表６】

表６ バンクオペレーション

| O[1:0] | ニモニック | オペレーション | サイクル |
|---|---|---|---|
| 00 | IDLE | アイドル | 1 |
| 01 | PP | Precharge page(ページプリチャージ) | 3 |
| 10 | AP | Access Page(ページアクセス) | 5 |
| 11 | CP | Change Page(ページ変更) | 5 |

【０１０６】図２６を参照し、ビット$D_{1:0}$が、共通バス１１４を通じて接続される４つの可能なデュアルピクセル３DRAM装置１１０のうち１つを選択する。ビット$B_{4:0}$が、単一のデュアルピクセル３DRAM装置１１０内の３２の可能なDRAMバンク１５８のうち１つを選択する。ビット$P_{9:0}$が、DRAMバンク１５８内の１０２４個の可能なページ１６２のうち１つを選択する。

【０１０７】IDLEオペレーションは、１サイクルの間何もしない。図２６および図２７を参照し、Precharge Page（PP：ページプリチャージ）オペレーションは、装置$D_{1:0}$のバンク$B_{4:0}$をプリチャージし、送信に３サイクル、実行に８サイクルかかる。

【０１０８】図２６および図２８を参照し、Access Page（AP：ページアクセス）オペレーションは、装置$D_{1:0}$のバンク$B_{4:0}$のページ$P_{9:0}$にアクセスし、送信に５サイクル、実行に８サイクルかかる。DRAMバンク１５８は、前もってプリチャージされていなければならない。

【０１０９】図２６および図２９を参照し、Change Page（CP：ページ変更）オペレーションは、Access Pageオペレーションと、その後に続く同じ装置の同じバンクへのPrecharge Pageオペレーションとを組合せたものである。

【０１１０】1.5.2 グローバルバスオペレーション
グローバルバスオペレーションの転送は、帯域幅が２００メガバイト/秒の専用２ピン制御チャネル１５４により管理される。グローバルバスオペレーションは、任意のクロックの立下がり端において送信を開始できる。グローバルバス転送には４クロック必要なので、コマンド送信にも４クロックかかり得る。

【０１１１】図３０から図３４を参照し、ビット$D_{1:0}$は、共通バス１１４に繋がっている４つの可能なデュアルピクセル３DRAM装置１１０のうち1つを選択する。ビット$B_{4:0}$は、装置１１０内の３２の可能なDRAMバンク１５８のうち１つを選択する。ビット$C_{2:0}$は、選択されたDRAMバンク１５８内の８つの可能なキャッシュラインバッファ１６０のうち１つを選択する。ビット$L_{3:0}$は、SRAMピクセルバッファ１１８内の１６の可能なキャッシュライン２２４のうち１つを選択する。

【０１１２】次の表７は、一実施例において規定されるグローバルバスオペレーションのリストである。

【０１１３】
【表７】

39 40

表７　グローバルバスオペレーション

| O[2:0] | ニモニック | オペレーション | サイクル |
|---|---|---|---|
| 000 | IDLE | アイドル | 1 |
| 001 | - | 予約済 | - |
| 010 | RL | Read Cache Line(キャッシュライン読出) | 4 |
| 011 | - | 予約済 | - |
| 100 | WL | Write Cache Line(キャッシュライン書込) | 4 |
| 101 | ML | Masked Write Cache Line (キャッシュラインのマスク書込) | 4 |
| 110 | FL | Flash Masked Write Cache Line (キャッシュラインのフラッシュマスク書込) | 4 |
| 111 | CL | Change Cache Line (キャッシュライン変更) | 4 |

【０１１４】図３０を参照し、IDLEオペレーションは、１サイクルの間何もしない。Read Cache Line（RL）、Write Cache Line（WL）、Masked Cache Line（ML）およびChange Cache Line（CL）のコマンドはすべて、図３１に示すように制御（RQ）ピンにおいて同一のフォーマットを有する。図３２に、Fast Fill CacheLine（FL：キャッシュラインの高速フィル）コマンドを示す。

【０１１５】図３３を参照し、Read Cache Line（RL）オペレーションは、装置$D_{1:0}$にのみ与えられ、DRAMバンク１５８、$B_{4:0}$からパイプラインレジスタ１２７へキャッシュラインバッファ１６０、$C_{2:0}$をコピーする。次に、SRAMピクセルバッファ１１８のライン$L_{3:0}$へパイプラインレジスタ値を書込む。このオペレーションは、送信に４サイクル、各データ転送を行なうのに4サイクルかかる。SRAMピクセルバッファ１１８内の各ラインは、そのラインがどこから来たかを示すバンクおよびコラムタグ２３０を有する。バンクおよびコラムタグ２３０は、SARMピクセルバッファ１１８への書込転送の間にRLオペレーションによってセットされる。

【０１１６】図３４を参照し、Write Cache Line（WL）オペレーションは、装置$D_{1:0}$にのみ与えられ、SRAMピクセルバッファ１１８からパイプラインレジスタ１３７へキャッシュライン２２４、$L_{3:0}$をコピーする。パイプラインレジスタデータは、次にDRAMバンク１５８、$B_{3:0}$のコラム$C_{2:0}$へコピーされる。バンクおよびコラムタグ２３０は無視される。このオペレーションは、送信に４サイクル、各データ転送を行なうのに４サイクルかかる。

【０１１７】図３５を参照し、Masked Write Cache Line（ML）オペレーションは、装置$D_{1:0}$にのみ与えられ、SRAMピクセルバッファ１１８からパイプラインレジスタへキャッシュライン$L_{2:0}$をコピーする。次に、パイプラインレジスタのデータは、バンク$B_{3:0}$のコラム$C_{2:0}$へコピーされる。バンクおよびコラムタグは無視される。このオペレーションは、送信に４サイクル、各データ転送を行なうのに４サイクルかかる。プレーンマスクレジスタおよびバイトマスクレジスタは両方とも、１０２４ビットに拡張され、キャッシュラインがセンスアンプに書込まれるときビットごとの書込イネーブルとして使用される。

【０１１８】図３６を参照し、Flash Masked Write Cache Line（FL：キャッシュラインのフラッシュマスク書込）オペレーションは、パイプラインデータおよびマスクが１つのコラムにだけではなく４つのコラムに書込まれるという点を除いてはMLと同様である。このオペレーションは、装置$D_{1:0}$にのみ与えられ、SRAMからパイプラインレジスタへキャッシュライン$L_{2:0}$をコピーする。次に、パイプラインレジスタのデータは、バンク$B_{3:0}$のコラム０～３または４～７のいずれかへコピーされる。バンクおよびコラムタグは無視される。このオペレーションは、送信に４サイクル、各データ転送を行なうのに４サイクルかかる。プレーンマスクレジスタおよびバイトマスクレジスタは両方とも、１０２４ビットに拡張され、キャッシュラインがセンスアンプに書込まれるときビットごとの書込イネーブルとして使用される。

【０１１９】図３７を参照し、Change Cache Line（CL）オペレーションが、装置$D_{1:0}$にのみ与えられ、同時にRead Cache Line（RL）オペレーションおよびWrite Cache Line（WL）オペレーションを行なう。オペレーションからのバンクおよびコラムフィールドが、RLコマンドを制御する。キャッシュライン２２４からのバンクおよびコラムタグ２３０が、WLコマンドを制御する。このオペレーションは、送信に４サイクル、各データ転送を行なうのに４サイクルかかる。

【０１２０】1.5.3 ピクセルALUオペレーション

ピクセルALUオペレーションは、帯域幅が４００メガバイト/秒の専用４ピン制御チャネル１５２を通じて送信される。ピクセルALUオペレーションは、任意のクロックの立下がり端で送信を開始できる。IDLE以外のオペレーションは、制御ピンを介する送信に２サイクル必要とする。各ピクセルALUオペレーションは、データピン（D

QA[8:0]およびDQB[8:0]）を介する２サイクルでのデータ７２ビットの転送または３サイクルでのデータ１０８ビットの転送を制御する。各オペレーションの~2/3ビットはいくつのサイクルが必要であるかを示す。データ転送に3サイクル必要とするピクセルALUオペレーション＊は、その後にIDLEサイクルが続かねばならない。次の表８に挙げるオペレーションは、一実施例について規定されるピクセルALUオペレーションである。

【０１２１】

【表８】

表８　ピクセルALUオペレーション

| O[3:0] | ニモニック | オペレーション | サイクル |
|---|---|---|---|
| 0000 | IDLE | アイドル | 1 |
| 0001 | - | 予約済 | |
| 0010 | - | 予約済 | 2 |
| 0011 | - | 予約済 | |
| 0100 | RREG | Read Register(レジスタ読出) | 2 |
| 0101 | - | 予約済 | |
| 0110 | WREG | Write Register(レジスタ書込) | 2 |
| 0111 | BREG | Broadcast Register (レジスタブロードキャスト) | 2 |
| 1000 | RDAT | Read Data(データ読出) | 2 |
| 1001 | - | 予約済 | |
| 1010 | WDAT | Write Data(データ書込) | 2 |
| 1011 | BDAT | Broadcast Data (データブロードキャスト) | 2 |
| 1100 | RPIX | Read Pixel(ピクセル読出) | 2または3 |
| 1101 | - | 予約済 | |
| 1110 | SPIX | Single Pixel(シングル・ピクセル) | 2 |
| 1111 | DPIX | Dual Pixel(デュアル・ピクセル) | 2または3 |

【０１２２】ビット$O_{3:0}$は、オペレーションのタイプを指定する。ビット$D_{1:0}$は、共通バス１１４に繋がっている４つのデュアルピクセル３DRAM装置１１０のうち１つを選択する。ブロードキャストオペレーションにおいては、$D_{1:0}$は無視される。ビット$L_{2:0}$は、SRAMピクセルバッファ１１８内の８つのキャッシュライン２２４のうち１つを選択する。ビット$P_{3:0}$は、キャッシュライン２２４内の１６個のピクセルのうち１つを選択する。$P_{3:0}$の解釈は、オペレーションのタイプおよび現在のピクセルのデプスに依存して幅がある。ビット$R_{7:0}$は、レジスタのアドレスを指定するためレジスタオペレーションにより使用される。

【０１２３】図３８を参照し、IDLEは、１サイクルの間何もしない。図３９を参照し、Read Data（RDAT：データ読出）オペレーションは、指定された装置１１０のSRAMピクセルバッファ１１８の指定されたライン２２４の指定されたピクセルから生データ６４ビットを読出す。このオペレーションは、いかなるレジスタのセッティングによっても影響されない。

【０１２４】Write Data(WDAT：データ書込）オペレーションは、指定された装置１１０のSRAMピクセルバッファ１１８の指定されたライン２２４の指定されたピクセルへ生データ６４ビットを書込む。バイトごとの書込イネーブルが、ピン$DQA_8$および$DQB_8$を通じて送信される。このオペレーションはいかなるレジスタのセッティングによっても影響されない。

【０１２５】Broadcast Data（BDAT：データブロードキャスト）オペレーションは、すべてのデュアルピクセル３DRAM装置１１０のSRAMピクセルバッファ１１８の指定されたライン２２４の指定されたピクセルへ生データ６４ビットをブロードキャストする。バイトごとの書込イネーブルが、ピン$DQA_8$および$DQB_8$を通じて送信される。このオペレーションはいかなるレジスタのセッティングによっても影響されない。

【０１２６】図４０を参照し、Read Register（RREG：レジスタ読出）オペレーションは、指定された装置１１０からレジスタ値を読出す。

【０１２７】Write Register（WREG：レジスタ書込）オペレーションは、レジスタ値を指定された装置１１０へ書込む。バイトごとの書込イネーブルが、ピン$DQA_8$および$DQB_8$を通じて送信される。

【０１２８】Broadcast Register（BREG：レジスタブロードキャスト）オペレーションは、レジスタ値をすべての装置１１０へブロードキャストする。バイトごとの書込イネーブルが、ピン$DQA_8$および$DQB_8$を通じて送信される。

【０１２９】図４１を参照し、Read Pixel (RPIX) オペレーションは、表示リフレッシュのためにパックされたピクセルを読出す。

【０１３０】Single Pixel (SPIX) オペレーションは、単一のソースピクセルを既にSRAMピクセルバッファ１１８内にあるピクセルデータとマージする。

【０１３１】Dual Pixel (DPIX) オペレーションは、２つの隣り合ったソースピクセルを既にSRAMピクセルバッファ１１８内にあるピクセルデータとマージする。

【０１３２】デュアルピクセル３DRAMチップ１１０の一実施例は、ダーティタグを使用する。この実施例においては、Write Tag (WTAG：タグ書込) オペレーションは、バイトマスクデータ６４ビットを１２８ビットのマスクに拡張し、これは指定された装置のダーティタグ２２６の指定されたライン２２４の上位半分または下位半分に書込まれる。ラインに対するダーティタグ２２６は、グローバルバスを通じてDRAMアレイにラインを書込むのにかかる時間の量と一致して、２サイクルで書込むことができる。

【０１３３】図４２は、デュアルピクセル３DRAM装置１１０が各サイクルにおいてデータ３６ビットを転送できる一実施例を示す。データの２クロックまたは３クロックが、各ピクセルALUオペレーションに関連付けられる。

【０１３４】図４３を参照し、Read Data (RDAT) オペレーションおよびRead Pixel (RPIX) オペレーションは、SRAMピクセルバッファ１１８ならびにピクセルALU１２０および１２１のいくつかを用いる。ピクセルALU１２０および１２１が使用されず、SRAMピクセルバッファ１１８には何も書込まれない。図４３は、３サイクルRPIXが後に続く２サイクルRDATを示す。

【０１３５】図４４に示すように、Write Data (WDAT) オペレーション、Broadcast Data (BDAT) オペレーション、Write Register (WREG) オペレーションおよびBroadcast Register (BREG) オペレーションは、DQピン上でデータを受信し、ピクセルALU１２０および１２１を通じてデータを送り、適当な段でデータをラッチする。上述のオペレーションのいずれも3サイクル転送による利益を受けない。

【０１３６】Single Pixel(SPIX) オペレーションおよびDual Pixel (DPIX) オペレーションは、SRAMピクセルバッファ１１８ならびにピクセルALU１２０および１２１を十分に利用する。これらの２つのオペレーションは、SRAMピクセルバッファ１１８からデータを読出しアンパックし、DQピンからデータを受信し、ピクセルALU１２０および１２１内のデータの組を両方組合せ、再びSRAMピクセルバッファ１１８へ結果を書込む。図４５は２サイクルでのSPIX転送を示し、図４６は３サイクルでのDPIX転送を示す。

【０１３７】1.6 オペレーションタイミング

上に説明したオペレーションの多くのタイミングの例を図４７から図５３に示す。

【０１３８】図４７は、複合２サイクル読出および２サイクル書込オペレーションの図である。

【０１３９】図４８は、複合２サイクル読出および３サイクル書込オペレーションの図である。

【０１４０】図４９は、複合３サイクル読出および２サイクル書込オペレーションの図である。

【０１４１】図５０は、複合３サイクル読出および３サイクル書込オペレーションの図である。

【０１４２】図５１は、４つの２サイクル読出オペレーションを行なうために必要とされるすべてのオペレーションの図である。

【０１４３】図５２および図５３は、８つの２サイクルDual Pixel (DPIX) オペレーションを行なうため必要とされるすべてのオペレーションを示す。DPIXオペレーション４～７は、DPIXオペレーション０～３に対するバンクとは異なるバンク１５８へのものである。

【０１４４】1.7 レジスタ

RREGオペレーション、WREGオペレーションおよびBREGオペレーションは、１２８ワード×６４ビットレジスタアドレススペースに対応したものである。ピクセルALU１２０および１２１を再プログラムするため必要とされるサイクルの数を最小限にするため、レジスタの幅を利用して効率化が図られる。次の表９は、デュアルピクセル３DRAMチップ１１０において用いられるレジスタを識別するものである。図５４は、表９に挙げられるいくつかのレジスタのデータフォーマットを示す。影をつけた部分はこれらのレジスタ用に予約されたフィールドを表わす。

【０１４５】

【表９】

表９ レジスタマップ

| $R_{7:0}$ | 名前 | ニモニック | リセット値 | アクト |
|---|---|---|---|---|
| 0 | Identification | ID | N/A | R |
| 1 | Feature Enable | FE | 0x0000 0000 0000 0000 | R/W |
| 2 | Pixel Config | PC | 0x0000 0000 0000 0000 | R/W |
| 3 | Stencil Depth Config | SDC | 0x0000 0000 0000 0000 | R/W |
| 5-4 | ColorOp[1:0] | CO | 0x0000 0000 0000 0000 | R/W |
| 6 | Constant Color | CC | 0x0000 0000 0000 0000 | R/W |
| 13-7 | 予約済 | - | - | - |
| 15-14 | Byte Mask [1:0] | BM | 0xFFFF FFFF FFFF FFFF | R/W |
| 23-16 | Plane Mask [7:0] | PM | 0xFFF FFFF FFFF FFFF | R/W |
| 63-23 | 予約済 | - | - | - |
| 67-64 | ColorWIDLUT [3:0] | CWL | 0x0000 0000 0000 0000 | R/W |
| 71-68 | OverlayWIDLUT [3:0] | OWL | 0x0000 0000 0000 0000 | R/W |
| 72 | Display Config | DC | 0x0000 0000 0000 0000 | R/W |
| 127-73 | 予約済 | - | - | - |

【０１４６】1.7.1 識別
この読出専用レジスタは、チップのマスクステッピング、バージョン、部品番号および製造者を識別する。
【０１４７】1.7.2 FeatureEnable
このレジスタは、デュアルピクセル３DRAMの将来のバージョンにおける新しい機能を可能または不能にする。デュアルピクセル３DRAMの初期のバージョンにおいては、これは、0x0000#0000#0000#0000にリセットされ、他のいかなる値にもセットされるべきではない。

＊【０１４８】1.7.3 PixelConfig
図５５は、PixelConfigレジスタのデータフィールドフォーマットを示す。このレジスタは、ピクセルのデプスおよびピクセルの詳細なフォーマットを決定する。影つきで図示される予約されたフィールドは、将来の互換性を確保するために、０にセットされなければならない。表１０は、レジスタのデータフィールドを説明する。
【０１４９】
【表１０】

表10 PixelConfig レジスタフィールド

| フィールド | 幅 | 記述 |
|---|---|---|
| PixelSize | 3 ビット | SRAM→PALU および PALU→SRAM のピクセルサイズ |
| ColorMode | 4 ビット | 色データがストアされる態様を決定する |
| BufferSelect | 2 ビット | A/B 色データを選択する |
| InputMode | 4 ビット | DQ→PALU ルーティングを制御する |
| DestinationFactor | 4 ビット | 行先ブレンド係数 |
| SourceFactor | 4 ビット | ソースブレンド係数 |
| WIDMask | 8 ビット | WID マスク値 |
| WIDRef | 8 ビット | WID 基準値 |
| WIDFunc | 3 ビット | WID 比較オペレーション |

【０１５０】PixelSizeフィールドは、SPIXオペレーションおよびDPIXオペレーションの間にピクセルサイズを選択するため、ピクセルALUからSRAMへのフォーマッタ１４０およびSRAMからピクセルALUへのフォーマッタ１４４により使用される。
【０１５１】
【表１１】

表11 ピクセルサイズ

| PS[2:0] | ピクセルサイズ |
|---|---|
| 0 | 8 ビット, 16 ビット, 32 ビット |
| 1 | 64 ビット |
| 2 | 128 ビット |
| 3 | 256 ビット |
| 4 | 512 ビット |

【０１５２】ColorModeフィールドは、フォーマッタ１４０および１４４用の色データフォーマットを指定する。表１２は色モードフォーマットを説明する。
【０１５３】
【表１２】

表12 ColorMode フィールド

| エンコーディング | 色フォーマット | | | |
|---|---|---|---|---|
| | アルファ | 赤 | 緑 | 青 |
| 0 | 8 | 8 | 8 | 8 |
| 1 | 8 | 0 | 0 | 0 |
| 2 | 2 | 10 | 10 | 10 |
| 3 | 10 | 10 | 10 | 10 |
| 4 | 4 | 4 | 4 | 4 |
| 5 | 8 | 8 | 0 | 0 |
| 6 | 0 | 5 | 6 | 5 |
| 7 | 1 | 5 | 5 | 5 |

【０１５４】BufferSelectフィールドは、ピクセルALU１２０および１２１とSRAMピクセルバッファ１１８との間でデータをフォーマット化するフォーマッタ１４０お

よび１４４のための色バッファを選択する。BufferSelect[0]は、３２ビットワード内の色の下位１６ビットと上位１６ビットとのいずれかを選択する。もし、色が３２ビットまたは４０ビットであれば、BufferSelect[0]は無効である。BufferSelect[1]は、ピクセル内に色６４ビットまたは８０ビットがあるとき、色データの下位もしくは上位３２ビットまたは下位もしくは上位４０ビットを選択する。BufferSelect[1]は、もしPixelSizeが＊６４ビットであれば無効である。

【０１５５】InputModeは、入力データフォーマッタ１３０を制御する。ソース係数フィールドおよび行き先係数フィールドは、表１３に説明するようにエンコードされる。

【０１５６】

【表１３】

表13 ブレンドオペレーションフィールド

| エンコーディング | 係数 アルファ | 赤 | 緑 | 青 | OpenGL |
|---|---|---|---|---|---|
| 0 | 0 | | | | GL_ZERO |
| 1 | 1 | | | | GL_ONE |
| 2 | 1 | | min (Sα,1-Dα) | | GL_SRC_ALPHA_SATURATE |
| 4 | Sα | Sr | Sg | Sb | GL_SRC_COLOR |
| 5 | 1-Sα | 1-Sr | 1-Sg | 1-Sb | GL_ONE_MINUS_SRC_COLOR |
| 6 | Sα | | | | GL_SRC_ALPHA |
| 7 | 1-Sα | | | | GL_ONE_MINUS_SRC_ALPHA |
| 8 | Dα | Dr | Dg | Db | GL_DST_COLOR |
| 9 | 1-Dα | 1-Dr | 1-Dg | 1-Db | GL_ONE_MINUS_DST_COLOR |
| 10 | Dα | | | | GL_DST_ALPHA |
| 11 | 1-Dα | | | | GL_ONE_MINUS_DST_ALPHA |
| 12 | Cα | Cr | Cg | Cb | GL_CONSTANT_COLOR |
| 13 | 1-Cα | 1-Cr | 1-Cg | 1-Cb | GL_ONE_MINUS_CONSTANT_COLOR |
| 14 | Cα | | | | GL_CONSTANT_ALPHA |
| 15 | 1-Cα | | | | GL_ONE_MINUS_CONSTANT_ALPHA |

【０１５７】WIDFuncフィールド、DepthFuncフィールドおよびStencilFuncフィールドは、表１４に説明するようにエンコードされる。

※【０１５８】

【表１４】

※

表14 WIDFunc, DepthFunc, およびStencilFuncフィールド

| DepthFunc | 条件 | OpenGL |
|---|---|---|
| 0 | Pass | GL_ALWAYS |
| 1 | Source > Destination | GL_GREATER |
| 2 | Source == Destination | GL_EQUAL |
| 3 | Source >= Destination | GL_GEQUAL |
| 4 | Fail | GL_NEVER |
| 5 | Source <= Destination | GL_LEQUAL |
| 6 | Source != Destination | GL_NOTEQUAL |
| 7 | Source < Destination | GL_LESS |

【０１５９】1.7.4 StencilDepthConfig

図５６は、StencilDepthConfigレジスタのデータフィールドフォーマットを示す。このレジスタは、ステンシルユニット１７０およびデプスユニット１６８を制御する。１６ビットのデプスマスクフィールド、３ビットのデプス比較ファンクションフィールド、および１ビットのDepthLoadフィールドが、デプスユニット１７０を制御する。StencilMaskフィールドが、ステンシル比較オペレーションにおいてどのビットが関係するかを決定する。行き先データがStencilRef値と比較される。StencilFuncが、どのように行き先値と基準値とが比較されるかを指定する。「StencilOp」で始まるフィールドが、新しいステンシルデータをどのように計算するかを決定する。影つきで図示される予約されたフィールドは、将来の互換性を確保するために、０にセットされなければならない。

【０１６０】

【表１５】

表15 StencilDepthConfig レジスタ

| フィールド | 幅 | 記 述 |
|---|---|---|
| DepthMask | 16 ビット | デプスのどのビットが比較されるかを制御する |
| DepthFunc | 3 ビット | 比較オペレーションを選択する |
| DepthLoad | 1 ビット | DPIX の間に入力アキュムレータにロードする |
| StencilMask | 8 ビット | ステンシルマスク値 |
| SencilRef | 8 ビット | ステンシル基準値 |
| StencilFunc | 3 ビット | ステンシル比較オペレーション |
| StencilOpFail | 3 ビット | もしステンシルテストがフェールであればステンシルオペレーション |
| StencilOpZfail | 3 ビット | もしステンシルテストがパスでデプステストがフェールであればステンシルオペレーション |
| StencilOpZpass | 3 ビット | ステンシルテストがパスでデプステストがフェールであればステンシルオペレーション |

【０１６１】DepthFuncフィールドおよびStencilFuncフィールドは、前掲の表１４によりエンコードされる。

【０１６２】「StencilOp」で始まるフィールドは、次＊の表１６に説明するようにエンコードされる。

【０１６３】

【表１６】

表16 StencilOp で始まるフィールド

| StencilOp | オペレーション | OpenGL |
|---|---|---|
| 0 | Destination | GL_KEEP |
| 1 | 0 | GL_ZERO |
| 2 | Reference | GL_REPLACE |
| 3 | Destination | GL_INVERT |
| 4 | Saturate(Destination+1) | GL_INCR |
| 5 | Saturate(Destination-1) | GL_DECR |
| 6 | Destination+1 | GL_INCRWRAP |
| 7 | Destination-1 | GL_DECWRAP |

【０１６４】1.7.5 ColorOP[0]

図５７は、ColorOP[0]レジスタフィールドを示す。ColorOP[0]レジスタは、アルファ、赤、緑および青ROPおよびブレンドユニット１７４および１７６の個々のための制御フィールドを有する。影つきで図示される予約されたフィールドは、将来の互換性を確保するため、０にセットされなければならない。

【０１６５】

【表１７】

表17 ColorOp[0] レジスタ

| フィールド | 幅 | 記 述 |
|---|---|---|
| AlphaLogicOp | 8 ビット | アルファ論理オペレーション |
| AlphaBlendOp | 3 ビット | アルファブレンドオペレーション |
| RedLogicOp | 8 ビット | 赤論理オペレーション |
| RedBlendOp | 3 ビット | 赤ブレンドオペレーション |
| GreenLogicOp | 8 ビット | 緑論理オペレーション |
| GreenBlendOp | 3 ビット | 緑ブレンドオペレーション |
| BlueLogicOp | 8 ビット | 青論理オペレーション |
| BlueBlendOp | 3 ビット | 青ブレンドオペレーション |

【０１６６】論理オペレーションフィールドは、ソース色、行き先色およびパターン色の２５６個のビットごとのブール演算のうち１つを選択するため、Microsoft Windows™ 規約によってエンコードされる。パターンデータは、ConstantColorレジスタから来る。

【０１６７】ブレンドオペレーションフィールドは、表１８で説明するようにエンコードされる。

【０１６８】

【表１８】

51 52

表18 ブレンドオペレーションフィールド

| BlendOp | オペレーション | OpenGL |
|---|---|---|
| 0 | LogicOp (Sc, Dc, Pattern) | GL_COLOR_LOGIC_OP |
| 1 | Sc*Sf+Dc*Df | GL_FUNC_ADD |
| 2 | Sc*Sf-Dc*Df | GL_FUNC_SUBTRACT |
| 3 | Dc*Df-Sc*Sf | GL_FUNC_REVERSE_SUBTRACT |
| 4 | min(Sc, Dc) | GL_MIN |
| 5 | max(Sc, Dc) | GL_MAX |

【０１６９】1.7.6 ColorOP[1]

図５８は、ColorOP[1]レジスタのデータフィールドフォーマットを示す。ColorOP[1]レジスタは、アルファ、赤、緑および青ROPおよびブレンドユニット１７６個々のための制御フィールドを有する。影つきで図示される＊予約されたフィールドは、将来の互換性を確保するため０にセットされねばならない。表１９は、ColoROP[1]レジスタのフィールドを説明する。

【０１７０】

【表１９】

表19 ColorOp[1] レジスタ

| フィールド | 幅 | 記 述 |
|---|---|---|
| AlphaSize | 4 ビット | アルファデータのサイズ |
| AlphaLoad | 1 ビット | DPIX の間にアルファアキュムレータにロードする |
| AlphaBlendEnable | 1 ビット | ブレンド用にアルファ入力をフォーマット化する |
| AlphaDitherEnable | 1 ビット | アルファのディザリングを可能化する |
| RedSize | 4 ビット | 赤データのサイズ |
| RedLoad | 1 ビット | DPIX の間に赤アキュムレータにロードする |
| RedBlendEnable | 1 ビット | ブレンド用に赤入力をフォーマット化する |
| RedDitherEnable | 1 ビット | 赤のディザリングを可能化する |
| GreenSize | 4 ビット | 緑データのサイズ |
| GreenLoad | 1 ビット | DPIX の間に緑アキュムレータにロードする |
| GreenBlendEnable | 1 ビット | ブレンド用に緑入力をフォーマット化する |
| GreenDitherEnable | 1 ビット | 緑のディザリングを可能化する |
| BlueSize | 4 ビット | 青データのサイズ |
| BlueLoad | 1 ビット | DPIX の間に青アキュムレータにロードする |
| BlueBlendEnable | 1 ビット | ブレンド用に青入力をフォーマット化する |
| BlueDitherEnable | 1 ビット | 青のディザリングを可能化する |

【０１７１】成分サイズファイルは表２０に説明するようにエンコードされる。

【０１７２】

【表２０】

表20 サイズフィールドのエンコーディング

| エンコーディング | サイズ |
|---|---|
| 1 | 1 ビット |
| 2 | 2 ビット |
| 3 | 3 ビット |
| 4 | 4 ビット |
| 5 | 5 ビット |
| 6 | 6 ビット |
| 7 | 7 ビット |
| 8 | 8 ビット |
| 9 | 9 ビット |
| 10 | 10 ビット |

【０１７３】1.7.7 ConstantColor

図５９は、ConstantColorレジスタのデータフィールドフォーマットを示す。ConstantColorレジスタは、論理オペレーション用のパターンデータまたはブレンドオペレーション用の定数データのいずれかとして使用される。影つきで図示される予約されたフィールドは、将来の互換性を確保するためセットされねばならない。表２１は、ConstantColorレジスタのフィールドを説明する。

【０１７４】

【表２１】

表21 Constant Color レジスタ

| フィールド | 幅 | 記 述 |
|---|---|---|
| AlphaConstant | 10 ビット | アルファ定数データ |
| RedConstant | 10 ビット | 赤定数データ |
| GreenConstant | 10 ビット | 緑定数データ |
| BlueConstant | 10 ビット | 青定数データ |

【０１７５】1.7.8 Byte Mask[1:0]

これら２つのレジスタは、１２８ビットのバイトマスクへの読出/書込アクセスを可能にする。バイトマスクは、MLオペレーションおよびFLオペレーションに影響する。

【０１７６】1.7.9 Plane Mask[7:0]

これら8つのレジスタは、５１２ビットのプレーンマスクへの読出/書込アクセスを可能にする。プレーンマスクは、MLオペレーション、FLオペレーション、SPIXオペレーションおよびDPIXオペレーションに影響する。

【０１７７】1.7.10 ColorWIDLUT[3:0]

これら４つのレジスタは、Aバッファ(0)またはBバッフ

ァ(1)のαRGB色データを選択するためウィンドウIDルックアップテーブルにロードする。表示リフレッシュの間に、８ビットのWIDは、選択ビットを生成するための２５６エントリルックアップテーブル（LUT）へのインデックスとなる。

【０１７８】1.7.11　OverlayWIDLUT[3:0]

これら４つのレジスタは、Aバッファ(0)またはBバッファ(1)のオーバーレイデータを選択するためウィンドウIDルックアップテーブルにロードする。表示リフレッシュの間、８ビットWIDは、選択ビットを生成するための２５６エントリーLUTへのインデックスとなる。

【０１７９】1.7.12　DisplayConfig

図６０は、DisplayConfigレジスタのデータフィールドフォーマットを示す。このレジスタは、RPIX命令の間にピクセルデータの処理を制御するため、３ビットのピクセルサイズフィールドを有する。影つきで図示される予約されたフィールドは、将来の互換性を確保するため０にセットされねばならない。PixelSizeフィールドは前掲の表１１によりエンコードされる。

【０１８０】1.8 高速領域クリア

MLオペレーション、FLオペレーション、APオペレーション、PPオペレーション、WREGオペレーションおよびBREGオペレーションは、領域を極めて迅速にクリアするため、あわせて使用することができる。ByteMask[1:0]レジスタは、キャッシュライン内の個々のピクセルを書込用に可能化または不能化するため使用できる。PlaneMask[7:0]レジスタは、書込用にピクセル成分を可能化または不能化するため使用できる。

【０１８１】もし、ByteMask[1:0]レジスタを使用して個々のピクセルをマスクする必要があるのであれば、APオペレーション、PPオペレーション、WREGオペレーションおよびMLオペレーションを使用しなければならない。いずれのピクセルもまだフィルされないうちに、１０２４ビットキャッシュライン中のすべてのピクセルが所望のクリア値にセットされ、５１２ビットPlaneMaskレジスタが、クリアされるべきピクセルのこれらのビットへの書込のみを可能化するようセットされる。次に、WREGオペレーションまたはBREGオペレーション２つを使用して、１２８ビットByteMaskレジスタが書込みされる。次に、MLオペレーションが、ByteMaskレジスタおよびPlaneMaskレジスタを使用して、キャッシュラインをセンスアンプに書込む。続くWREGおよびMLオペレーションシーケンスは、１０ナノ秒ごとに１２８バイトまたは１２．８ギガバイト/秒のピークフィル速度をサポートするよう重ね合わせることができる。８０メガビットの装置は、８１９マイクロ秒でフィルされ得る。

【０１８２】もしページ内のすべてのピクセルをクリアするのであれば、１２８ビットByteMaskレジスタはオール１にセットでき、WREG、MLオペレーションシーケンスの代わりにFLオペレーションシーケンスを使用できる。１０２４ビットキャッシュラインおよび５１２ビットPlaneMaskレジスタは前と同様セットされる。FLオペレーションごとにページの半分がクリアされ、したがって、ページ全体は２０ナノ秒でクリアされ得る。ピークフィル速度は、２０ナノ秒ごとに８×１２８バイトまたは５１．２ギガバイト/秒である。８０メガビットの装置は、２０５マイクロ秒でフィルされ得る。表２２は、すべてのピクセルサイズについてのピークフィル速度を示す。図６１および図６２は、上に説明した高速フィル（Fast Fill）および非常に高速のフィル（Really Fast Fill）のオペレーションシーケンスを示す。

【０１８３】

【表２２】

表22　高速領域クリアピーク速度

| ピクセルサイズ | ML フィル速度 | FL フィル速度 |
|---|---|---|
| 8 ビット | 12.8 GP/s | 51.2 GP/s |
| 16 ビット | 6.4 GP/s | 25.6 GP/s |
| 32 ビット | 3.2 GP/s | 12.8 GP/s |
| 64 ビット | 1.6 GP/s | 6.4 GP/s |
| 128 ビット | 800 MP/s | 3.2 GP/s |
| 256 ビット | 400 MP/s | 1.6 GP/s |
| 512 ビット | 200 MP/s | 800 MP/s |

【０１８４】2.0 データルーティング

デュアルピクセル３DRAMチップ１１０の多数の新規な特徴、および、そのチップを基礎とするグラフィクスシステムが、チップ１１０の４つのデータフォーマッタ１３０、１３４、１４０および１４４によって実装される。本開示のこのセクションを通じて、SRAMピクセルバッファ１１８を２つの個別の機能ブロックとして説明する。なぜなら、ここに開示する処理モードおよびルーティングモードの多くが２つのピクセルを同時に処理する動作モードに関連しているためである。したがって、SRAMピクセルバッファ１１８は、機能的に２つの部分に分割して考えることができるものと理解されたい。

【０１８５】2.1 入力データフォーマッタ

図６３を参照して、このセクションでは、I/Oバス１２４を介してピクセルALU１２０および１２１へと入来するデータのルーティングおよび処理について説明する。ピクセルALU書込動作は、２クロック内でデータの７２ビットを、または、３クロック内でデータの１０８ビットを、処理されるピクセルのフォーマットおよびサイズに応じて転送する。入来データはできるだけピンの近くの入力データデマルチプレクサ１２６を通過して、７２ビットまたは１０８ビットのいずれかとして並行にピクセルALUに与えられる。ピクセルALU１２０および１２１は、ソースデータとして以下の入力を有する：Alpha0[10:0], Red0[10:0], Green0[10:0], Blue0[10:0], Depth0[31:0], Alpha1[10:0], Red1[10:0], Green1[10:0],Blue1[10:0], および Depth1[31:0]。ルーティングは、４ビットレジスタフィールドによって制御される。

【０１８６】2.1.1 ピクセル圧縮

三次元ピクセルマージを行なう場合、レンダリングコントローラは、色およびデプス情報のみを送信すればよい。というのは、ステンシルおよびウィンドウID情報はレジスタ内にストアすることができるためである。三角形または表面パッチをレンダリングする場合、生成されたピクセルは通常、高レベルのコヒーレンシーを示す。このセクションでは、最小数のクロックサイクル内でデータピンを通じてピクセルペアを送信する、新規な無損失の圧縮方式について説明する。

【０１８７】この方式は、ピクセルペア間の差および、最も最近処理された(古い)ピクセルペアと入ってくる(新しい)ピクセルペアとの間の差を評価する。１対のピクセル間には、また、２対の連続して処理されるピクセルペア間には、高レベルのコヒーレンシーが存在することがしばしばあるため、入来するピクセルデータは時として、非常に少ない数のビットで表わすことができる場合がある。このような状況下では、送信すべきなのは古いピクセルペアと新しいピクセルペアとの差のみであって、この差は、新しいピクセルペアの最下位ビットで表わされる。最良の場合、すなわち、ピクセルペア間に高レベルのコヒーレンシーが見られる場合、２：１の圧縮比を達成することが可能であり、レンダリングコントローラ１０２からデュアルピクセル３DRAMチップ１１０への入力帯域幅を有効に２倍に増すことができる。中程度のコヒーレンシーが見られる場合には、４：３の圧縮比を達成することができる。コヒーレンシーがほとんど見られない場合、圧縮を行なうことはできない。

【０１８８】デュアルピクセル３DRAM１１０は、最も新しく送られてきたピクセルペアを、各デプス値につき３２ビットおよび４つの色成分の各々につき１０ビットで、レジスタの組内にストアする。したがって、１ピクセルあたりデータの７２ビットがレンダリングバス１１２を介して送信される。ピクセルはデュアルピクセル３DRAMチップ１１０ではしばしば対で処理されるため、１対のピクセルのための完全なデータは１４４ビットで表わされる。テクスチャマッピングは色値のコヒーレンシーを低下させる傾向にあるため、色値を圧縮することによって帯域幅をセーブすることはできないであろう。しかし、デプス値はほとんどの場合、高レベルのコヒーレンシーを示す。したがって、ここに開示する圧縮方式は、デプス値間の高度のコヒーレンシーを利用する。以下に、ピクセルデータを圧縮および伸長するためのアルゴリズムを説明する。

【０１８９】レンダリングコントローラ１０２およびデュアルピクセル３DRAMチップ１１０は両方とも、最も新しく送られてきたピクセルペアをストアしている。以下に列記するのは、「古い」ピクセル０および１を色（アルファ、赤、緑および青）成分およびデプス成分に分解したものである。

【０１９０】

【数３】

```
OA0, OR0, OG0, OB0, OZ0
OA1, OR1, OG1, OB1, OZ1
```

【０１９１】レンダリングコントローラ１０２はその通常のレンダリング処理の一部として、新しいピクセルペアを算出する。多くの場合、古いピクセル成分と新しいピクセル成分とは同様の値を有し、新しいピクセルペアもまた同様の値を有する。下に、「新しい」ピクセル０および１を色（アルファ、赤、緑および青）成分およびデプス成分に分解したものを示す。

【０１９２】

【数４】

```
NA0, NR0, NG0, NB0, NZ0
NA1, NR1, NG1, NB1, NZ1
```

【０１９３】もしピクセル成分が同様の値を有する場合、それらの差は小さく、成分自体よりも少ないビットで表わすことができる。ピクセル成分の差は、レンダリングコントローラ１０２によって以下の等式を使用して計算される。式中、頭に付された「D」は、差（difference）またはデルタ（delta）を表わす。

【０１９４】

【数５】

```
DA0 = NA0 - OA0; DA1 = (NA1 - OA1) - (NA0 - OA0);
DR0 = NR0 - OR0; DR1 = (NR1 - OR1) - (NR0 - OR0);
DG0 = NG0 - OG0; DG1 = (NG1 - OG1) - (NG0 - OG0);
DB0 = NB0 - OB0; DB1 = (NB1 - OB1) - (NB0 - OB0);
DZ0 = NZ0 - OZ0; DZ1 = (NZ1 - OZ1) - (NZ0 - OZ0);
```

【０１９５】算出された差のみが、レンダリングコントローラ１０２からデュアルピクセル３DRAMチップ１１０へと送信されるので、レンダリングバス１１２を介して送信されるピクセルあたりのビット数が低減される。新しいピクセル成分は、デュアルピクセル３DRAMチップ１１０上で入力データフォーマッタ１３０により、以下のように再生される。

【０１９６】

【数６】

```
NA0 = OA0 + DA0; NA1 = OA1 + DA0 + DA1
NR0 = OR0 + DR0; NR1 = OR1 + DR0 + DR1
NG0 = OG0 + DG0; NG1 = OG1 + DG0 + DG1
NB0 = OB0 + DB0; NB1 = OB1 + DB0 + DB1
NZ0 = OZ0 + DZ0; NZ1 = OZ1 + DZ0 + DZ1
```

【０１９７】図６４を参照して、入力データフォーマッ

タ１３０は、その伸長方式を３層で実現する。第１の層では、種々のフォーマットから１０個の差成分を抽出して、必要であればそれらを符号拡張する。第２の層では、それらの差を先のピクセル成分に加えて、新しいピクセル成分を再生する。第３の層では、新しいピクセル成分をフォーマット化する。色成分は、その最終幅にしたがって左にシフトされ、デプス値の上位１６ビットがマスクされる。

【０１９８】2.1.2 入力データフォーマット

下の表２３に記載するように、デュアルピクセル３DRAM１１０は、以下のような入力データフォーマットを有する。もしオペレーションがWDATまたはBDATである場合、入力フォーマットは、レジスタのプログラム方法にかかわらず、強制的にモード０にされる。オペレーションが＊SPIXである場合、入力フォーマットは強制的にモード７にされる。オペレーションがDPIXの場合、入力フォーマットは、InputModeレジスタフィールドに書込むことによって設定される。モード０および１は、奥行きが不要な二次元のピクセル更新の場合に使用される。モード２は、アンチエイリアシングオペレーション中に使用される。モード４および５は、三次元のピクセル更新に使用される。図６５は、レンダリングコントローラ１０２から入力データフォーマッタ１３０へとレンダリングバス１１２を介して送信されるデータについて、可能性のあるすべてのフォーマットを示す。

【０１９９】

【表２３】

表 23　入力データフォーマット内のビットフィールド割当て

| モード | オペレーション | サイクル | フィル速度 | アルファ | | 赤 | | 緑 | | 青 | | デプス | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | DA0 | DA1 | DR0 | DR1 | DG0 | DG1 | DB0 | DB1 | DZ0 | DZ1 |
| 0 | WDAT | 2 | 400 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | - | - |
| | BDAT DPIX | 3 | 267 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | - | - |
| 1 | DPIX | 2 | 400 | 2 | 2 | 10 | 10 | 10 | 10 | 10 | 10 | - | - |
| 2 | DPIX | 2 | 400 | - | - | - | - | - | - | - | - | 32 | 32 |
| 4 | DPIX | 2 | 400 | 6 | 4 | 6 | 4 | 6 | 4 | 6 | 4 | 18 | 14 |
| | | 3 | 267 | 8 | 6 | 8 | 6 | 8 | 6 | 8 | 6 | 28 | 24 |
| 5 | DPIX | 2 | 400 | 2 | 2 | 7 | 5 | 7 | 5 | 7 | 5 | 18 | 14 |
| | | 3 | 267 | 2 | 2 | 10 | 7 | 10 | 7 | 10 | 7 | 28 | 24 |
| 7 | SPIX | 2 | 200 | 10 | | 10 | | 10 | | 10 | | 32 | |

【０２００】2.1.3 アキュムレータ

図６６は、入力データフォーマッタのアキュムレータ層の実現を図示する。同じ１０ビット色アキュムレータ設計が、アルファ、赤、緑および青データを処理するのに使用される一方、同様に設計された３２ビットアキュムレータが、デプスデータを処理するのに使用される。WDAT、BDAT、およびSPIXオペレーション中には、差分データがアキュムレータ内に直接ロードされる。DPIXオペレーション中には、差分データのローディングはレジスタビットによって制御される。

【０２０１】2.1.4 最終フォーマッティング

図６７は、入力データフォーマッタ１３０の最終フォーマット層を図示する。色成分は、ROP/Blendユニット１６６によって使用されるであろうビット数にしたがって左にシフトされる。もし成分がブレンドされる場合、１が最下位ビットの右にアペンドされる。以下に示すのは、入力データフォーマッタ１３０内で処理の最終層の色およびデプスのフォーマッティングを行なう、２つのverilog関数である。

【０２０２】

【数７】

59 60

```
function [10:0] FormatColor;
   input [9:0] Data;   // Raw color component data
   input [3:0] Size;   // Size of color component
   input       Blend;  // Set if colors are to be blended
   begin
      casex(Size)
      4'b0001: FormatColor = {Data[  0], Blend, 9'b000000000}; // 1 bit
      4'b0010: FormatColor = {Data[1:0], Blend, 8'b00000000};  // 2 bits
      4'b0011: FormatColor = {Data[2:0], Blend, 7'b0000000};   // 3 bits
      4'b0100: FormatColor = {Data[3:0], Blend, 6'b000000};    // 4 bits
      4'b0101: FormatColor = {Data[4:0], Blend, 5'b00000};     // 5 bits
      4'b0110: FormatColor = {Data[5:0], Blend, 4'b0000};      // 6 bits
      4'b0111: FormatColor = {Data[6:0], Blend, 3'b000};       // 7 bits
      4'b1000: FormatColor = {Data[7:0], Blend, 2'b00};        // 8 bits
      4'b1001: FormatColor = {Data[8:0], Blend, 1'b0};         // 9 bits
      4'b1010: FormatColor = {Data[9:0], Blend};               // 10
bits
      default: FormatColor = 'bx;
      endcase
   end
endfunction
```

【０２０３】デプス値の上位１６ビットは、DepthMaskレジスタフィールドと、ビットごとに論理積をとられる。

【０２０４】【数８】

```
function [31:0] FormatDepth;
     input [31:0] Data; // Raw depth data
     input [15:0] Mask; // Mask
     begin
          FormatDepth    =    {Data[31:16]    &    Mask,
Data[15:0]};
     end
endfunction
```

【０２０５】2.2 出力データフォーマッタ
オペレーションがRDATである場合、またはオペレーションがRPIXであってDisplayConfigレジスタのPixelSizeフィールドが８、１６、３２ビットピクセルに設定されている場合、フォーマッタはオペレーションのP[3:0]ビットにしたがって、１０２４キャッシュラインからデータの６４ビットを選択する。ColorWIDLUTおよびOverlayWIDLUTレジスタはこの場合無視される。

【０２０６】オペレーションがRPIXであってDisplayConfigレジスタのPixelSizeフィールドが６４ビットピクセルに設定されている場合、フォーマッタはオペレーションのP[3:1]ビットによってアドレシングされた６４ビットピクセルのペアから８ビットWIDフィールドを抽出する。抽出されたWIDフィールドは、ColorWIDLUTへのインデックスとなって、色A/Bバッファセレクトのペアが生成される。抽出されたWIDフィールドは、OverlayWIDLUTへのインデックスとなって、１６／３２ビットセレクトのペアが生成される。

【０２０７】この１６／３２ビットセレクトは、DQピンを介して色データの全３２ビットを送信するか、DQピンを介して色データの１６ビットのみを送信するかを決定する。後者の場合、A/Bバッファセレクトが、DQピンを介して色データの上位１６ビットか下位１６ビットのどちらを送信するかを決定する。

【０２０８】オペレーションがRPIXであって、DisplayConfigレジスタのPixelSizeフィールドが１２８ビットピクセルに設定されている場合、フォーマッタはオペレーションのP[3:2]ビットによってアドレシングされる１２８ビットピクセルのペアから８ビットWIDフィールドを抽出する。抽出されたWIDフィールドは、ColorWIDLUTへのインデックスとなって、色A/Bバッファセレクトのペアが生成される。抽出されたWIDフィールドはOverlayWIDLUTへのインデックスとなって、オーバレイA/Bバッファセレクトのペアが生成される。

【０２０９】色A/Bバッファセレクトは、DQピンを介してA色バッファデータかB色バッファデータか、いずれを送信するかを決定する。オーバレイA/Bバッファセレクトは、DQピンを介してAまたはBのいずれのオーバレイデータを送信するかを決定する。

【０２１０】図６８を参照して、このセクションではSRAMピクセルバッファ１１８からレンダリングコントローラ１０２へと出ていくデータのルーティングおよび処理について説明する。出ていくデータは、SRAM出力データバス１３２を介して出力データフォーマッタ１３４へと送信される。この出力データフォーマッタ１３４は、デュアルピクセル３DRAMチップ１１０から送信されるピクセルフォーマットに応じて、種々のモードで動作する。モードは、モードレジスタによって設定される。

【０２１１】この経路は、RDATおよびRPIXオペレーションによって使用される。ここで、データの２５６ビットがSRAMピクセルバッファ１１８から読出され、その２５６ビットからデータの７２ビットまたは１０８ビットが抽出される。これらはその後、出力データマルチプレクサ１３６に送られて、チップ１１０からレンダリングコントローラ１０２へとレンダリングバス１１２を介して送信される。RDATオペレーションは、２サイクル内でデータの６４ビットを読出すのに対し、RPIXオペレーションは２サイクルまたは３サイクル内で１ピクセルから８ピクセルを読出す。

【０２１２】2.2.1 RDAT、RPIX（８ビット、１６ビット、３２ビットピクセル）オペレーション

図６９は、８ビット、１６ビット、および３２ビットピクセルフォーマットのためのRDATおよびRPIXオペレーションを図示する。このモードでは、１０２４ビットキャッシュラインから６４ビットが選択される。SRAMピクセルバッファ１１８が２５６ビットを提供し、これがマルチプレクスされて６４ビットとなる。

【０２１３】このモードでは、連続する６４ビットが１０２４ビットキャッシュラインから選択されて、出力データマルチプレクサ１３６に与えられる。

【０２１４】2.2.2 RPIX（６４ビットピクセル）オペレーション

図７１から図７４は、６４ビットRPIXオペレーションの種々のモードおよび局面を図示する。図７１に示したモードにおいては、連続する６４ビットのピクセルが２つ、P[0]は無視して、１０２４ビットキャッシュラインから選択される。

【０２１５】図７２に示したモードでは、P[0]は無視して、２つの連続６４ビットピクセルが１０２４ビットキャッシュラインから選択され、その後処理されて、出力データマルチプレクサ１３６に提示される。

【０２１６】偶数の６４ビットピクセルは、図７３に示すように処理されて、出力データマルチプレクサ１３６に対して３６ビットの出力が生成される。ウィンドウIDビットは、色データを１６ビットずつダブルバッファリングするか、３２ビットでシングルバッファリングするかを決定し、ダブルバッファリングする場合には、AバッファかBバッファのどちらを選択するかを決定する。ウィンドウIDの８ビットが、２５６ビットColorWIDLUTレジスタおよび２５６ビットOverlayWIDLUTレジスタへのインデックスとなって、セレクトビットが生成される。

【０２１７】奇数の６４ビットピクセルは、図７４に示すように処理されて、出力データマルチプレクサ１３６に対して３６ビットの出力が生成される。ウィンドウIDビットが、色データを１６ビットずつダブルバッファリングするか、３２ビットでシングルバッファリングするかを決定し、ダブルバッファリングする場合には、AバッファかBバッファのどちらを選択するかを決定する。



【０２１８】2.2.3 RPIX（９６ビットピクセル）オペレーション

図７５から図７８は、９６ビットRPIXオペレーションの種々のモードおよび局面を図示する。図７５に示すように、このモードでは、２つの連続９６ビットのピクセルが、P[0]を無視して、１０２４ビットキャッシュラインから選択される。

【０２１９】２つの９６ビットピクセルは図７６に示すように並行に処理されて、出力データマルチプレクサ１３６に対して２つの４８ビット出力が生成される。ウィンドウIDビットは、AバッファかBバッファのどちらを選択するかを決定する。図７７および図７８は、両ピクセルについてのシングルバッファリングされるオーバレイおよびウィンドウIDがどのように扱われるかを示す。図７８は、色データの下位３バイトのためのデータ経路を示す。

【０２２０】2.2.4 RPIX（１２８ビットピクセル）オペレーション

図７９から図８２は、１２８ビットピクセルのRPIXオペレーションの種々の局面を図示する。図７９に示されたモードにおいて、２つの連続１２８ビットのピクセルが、P[1:0]は無視して、１０２４ビットキャッシュラインから選択される。

【０２２１】図８０に示したモードにおいて、２つの連続１２８ビットピクセルが、P[1:0]は無視して、１０２４ビットキャッシュラインから選択され、処理されて、出力データマルチプレクサ１３６に与えられる。

【０２２２】偶数の１２８ビットピクセルは、図８１に示すように処理されて、出力データマルチプレクサ１３６に対して４８ビット出力が生成される。ウィンドウIDビットは、AまたはBバッファ色、および、AまたはBバッファオーバレイを選択する。

【０２２３】奇数の１２８ビットピクセルは、図８２に示されるように処理されて、出力データマルチプレクサ１３６に対して４８ビット出力が生成される。ウィンドウIDビットは、AまたはBバッファ色、および、AまたはBバッファオーバレイを選択する。

【０２２４】2.3 SRAMからピクセルALUへのルーティング

SRAMからピクセルALUへと行先データを送る。SRAMからデータの２５６ビットを読出して、以下のピクセルALU入力に適切なフィールドを送る。Alpha0[10:0],Red0[10:0], Green0[10:0], Blue0[10:0], Depth0[31:0], Stencil0[7:0], WID0[7:0], Alpha1[10:0], Red1[10:0], Green1[10:0], Blue1[10:0], Depth1[31:0], Stencil1[7:0],および WID1[7:0]。すべてのピクセルALU入力は、レジスタ値によってマスクされる。

【０２２５】SRAM→DQおよびSRAM→PALUルーティングブロックは、共通で保有される回路はどれでも共用できる

よう、決して同時に使用されることはない。

【０２２６】SRAM→PALUルーティングは、２段階で行なうことができる。第１段階では、ピクセルのサイズ、アドレス、およびA/B色バッファセレクトにしたがって、各ピクセルの３２ビット区分を選択する。第２段階では、A/Bセレクト、色モードおよび種々のマスクにしたがって、色、デプス、ステンシル、およびWIDフィールドをアンパックする。

【０２２７】2.3.1 ８、１６、および３２ビットピクセルのSRAM編成

これらのピクセルは、アルファ、赤、緑および青データのみを含む。各オペレーションは、SRAMの片方ずつ両方に均等に分割された、６４ビット分のピクセルを処理する。

【０２２８】2.3.2 ６４ビットピクセルのSRAM編成

各オペレーションは、SRAMの片方ずつ両方に存在する、２つのピクセルを処理することができる。各ピクセルの「ab」区分は色データを含み、「de」区分はデプス、ステンシル、オーバレイ、およびWIDデータを含む。

【０２２９】2.3.3 ９６ビットピクセルのSRAM編成

やはり、各オペレーションは２つのピクセルを処理することができるが、これらの区分はいくぶん交錯している。各ピクセルは、「a」および「b」の個別の色区分を有する。

【０２３０】2.3.4 １２８ビットピクセルのSRAM編成

各オペレーションは、SRAMの片方ずつ両方に存在する２つのピクセルを処理することができる。デプス値は対応する「d」区分にストアされており、ステンシル、オーバレイ、およびWIDは「e」区分にストアされている。

【０２３１】2.3.5 UnpackColors

UnpackColorsは、PixelSize、PixelAddress、およびBufferSelectに基づいて、SRAMから色データの６４ビットを選択する。

【０２３２】

【数９】

```
function [63:0] UnpackColors;
    input [255:0] Data;
    input [2:0] PixelSize;
    input [3:0] PixelAddress;
    input [1:0] BufferSelect;
    begin
        casex({PixelSize, PixelAddress, BufferSelect[1]})
        // 8, 16, 32 bit pixels
        8'b000_xx00_x: UnpackColors = { Data[159:128], Data[ 31: 0]
};
        8'b000_xx01_x: UnpackColors = { Data[191:160], Data[ 63:32]
};
        8'b000_xx10_x: UnpackColors = { Data[223:192], Data[ 95:64]
};
        8'b000_xx11_x: UnpackColors = { Data[255:224], Data[127:96]
};
        // 64 bit pixels
        8'b001_xx0x_x: UnpackColors = { Data[159:128], Data[ 31: 0]
};
        8'b001_xx1x_x: UnpackColors = { Data[223:192], Data[ 95:64]
};
        // 96 bit pixels
        8'b010_x00x_0: UnpackColors = { Data[159:128], Data[ 31: 0]
};
        8'b010_x00x_1: UnpackColors = { Data[191:160], Data[ 63:32]
};
        8'b010_x01x_0: UnpackColors = { Data[159:128], Data[ 95:64]
};
        8'b010_x01x_1: UnpackColors = { Data[191:160], Data[127:96]
};
        8'b010_x10x_0: UnpackColors = { {32{1'bx}}, Data[ 31: 0] };
        8'b010_x10x_1: UnpackColors = { {32{1'bx}}, Data[ 63:32] };
        // 128 bit pixels
        8'b011_xxxx_0: UnpackColors = { Data[159:128], Data[ 31: 0]
};
        8'b011_xxxx_1: UnpackColors = { Data[191:160], Data[ 63:32]
};
        default: UnpackColors = 'bx;
        endcase
    end
endfunction
```

【０２３３】2.3.6 UnpackDepths

UnpackDepthsは、PixelSizeおよびPixelAddressに基づ

65 66

いて、デプス、ステンシル、オーバレイ、およびWIDの６４ビットの値を選択する。

＊【０２３４】
＊ 【数１０】

```
function [63:0] UnpackDepths;
      input [255:0] Data;
      input [2:0] PixelSize;
      input [3:0] PixelAddress;
      begin
            casex({PixelSize, PixelAddress})
            // 64 bit pixels
            7'b001_xx0x: UnpackDepths = { Data[191:160], Data[ 63: 32]
};
            7'b001_xx1x: UnpackDepths = { Data[255:224], Data[127: 96]
};
            // 96 bit pixels
            7'b010_x0xx: UnpackDepths = { Data[223:192], Data[255:224]
};
            7'b010_x10x: UnpackDepths = { {32{1'bx}}, Data[ 95: 64] };
            // 128 bit pixels
            7'b011_xxxx: UnpackDepths = { Data[223:192], Data[ 95: 64]
};
            default: UnpackDepths = 'bx;
            endcase
      end
endfunction
```

【０２３５】2.3.7 UnpackExtras

UnpackExtrasは、PixelSizeおよびPixelAddressに基づいて、ステンシル、オーバレイ、およびWIDの６４ビットの値を選択する。UnpackDepthsおよびUnpackExtras※は、６４ビットピクセルおよび９６ビットピクセルに対して同じデータを返す。

【０２３６】

※ 【数１１】

```
function [63:0] UnpackExtras;
      input [255:0] Data;
      input [2:0] PixelSize;
      input [3:0] PixelAddress;
      begin
            casex({PixelSize, PixelAddress})
            // 64 bit pixels
            7'b001_xx0x: UnpackExtras = { Data[191:160], Data[ 63: 32]
};
            7'b001_xx1x: UnpackExtras = { Data[255:224], Data[127: 96]
};
            // 96 bit pixels
            7'b010_x0xx: UnpackExtras = { Data[223:192], Data[255:224]
};
            7'b010_x10x: UnpackExtras = { {32{1'bx}}, Data[ 95: 64] };
            // 128 bit pixels
            7'b011_xxxx: UnpackExtras = { Data[255:224], Data[127: 96]
};
            default: UnpackExtras = 'bx;
            endcase
      end
endfunction
```

【０２３７】2.3.8 UnpackAlpha、UnpackRed、UnpackGreen、UnpackBlue

これらの関数は、３２ビットの色区分からアルファ、赤、緑、および青データをアンパックする。UnpackAlphaはまた、３２ビットのエクストラ区分を必要とする。ColorModeおよびBufferSelectは、色のアンパック方法を決定する。

【０２３８】

【数１２】

67 68

```
function [10:0] UnpackAlpha;
      input [31:0] Color;
      input [31:0] Extra;
      input [ 3:0] ColorMode;
      input [ 1:0] BufferSelect;
      begin
      casex({ColorMode, BufferSelect[0]})
      5'b0000_x: UnpackAlpha = {Color[31:24], 3'b100};
      5'b0001_x: UnpackAlpha = {Extra[31:24], 3'b100};
      5'b0010_x: UnpackAlpha = {11{1'b1}};
      5'b0011_0: UnpackAlpha = {Extra[ 7: 0], Color[31:30], 1'b1};
      5'b0011_1: UnpackAlpha = {Extra[15: 8], Color[31:30], 1'b1};
      5'b0100_0: UnpackAlpha = {Color[15:12], 7'b1000000};
      5'b0100_1: UnpackAlpha = {Color[31:28], 7'b1000000};
      5'b0101_0: UnpackAlpha = {Color[15: 8], 3'b100};
      5'b0101_1: UnpackAlpha = {Color[31:24], 3'b100};
      5'b0110_x: UnpackAlpha = {11{1'b1}};
      5'b0111_0: UnpackAlpha = {11{Color[15]}};
      5'b0111_1: UnpackAlpha = {11{Color[31]}};
      default: UnpackAlpha = 'bx;
      endcase
      end
endfunction

function [10:0] UnpackRed;
      input [31:0] Color;
      input [ 3:0] ColorMode;
      input [ 1:0] BufferSelect;
      begin
      casex({ColorMode, BufferSelect[0]})
      5'b0000_x: UnpackRed = {Color[23:16], 3'b100};
      5'b0001_x: UnpackRed = {11{1'b1}};
      5'b001x_x: UnpackRed = {Color[29:20], 1'b1};
      5'b0100_0: UnpackRed = {Color[11: 8], 7'b1000000};
      5'b0100_1: UnpackRed = {Color[27:24], 7'b1000000};
      5'b0101_0: UnpackRed = {Color[ 7: 0], 3'b100};
```

【０２３９】 【数１３】

69 70

```
        5'b0101_1: UnpackRed = {Color[23:16], 3'b100};
        5'b0110_0: UnpackRed = {Color[15:11], 6'b100000};
        5'b0110_1: UnpackRed = {Color[31:27], 6'b100000};
        5'b0111_0: UnpackRed = {Color[14:10], 6'b100000};
        5'b0111_1: UnpackRed = {Color[30:26], 6'b100000};
        default: UnpackRed = 'bx;
        endcase
        end
    endfunction

    function [10:0] UnpackGreen;
        input [31:0] Color;
        input [ 3:0] ColorMode;
        input [ 1:0] BufferSelect;
        begin
        casex({ColorMode, BufferSelect[0]})
        5'b0000_x: UnpackGreen = {Color[15: 8], 3'b100};
        5'b0001_x: UnpackGreen = {11{1'b1}};
        5'b001x_x: UnpackGreen = {Color[19:10], 1'b1};
        5'b0100_0: UnpackGreen = {Color[ 4: 7], 7'b1000000};
        5'b0100_1: UnpackGreen = {Color[23:20], 7'b1000000};
        5'b0110_0: UnpackGreen = {Color[10: 5], 5'b10000};
        5'b0110_1: UnpackGreen = {Color[26:21], 5'b10000};
        5'b0111_0: UnpackGreen = {Color[ 9: 5], 6'b100000};
        5'b0111_1: UnpackGreen = {Color[25:21], 6'b100000};
        default: UnpackGreen = 'bx;
        endcase
        end
    endfunction

    function [10:0] UnpackBlue;
        input [31:0] Color;
        input [ 3:0] ColorMode;
        input [ 1:0] BufferSelect;
        begin
        casex({ColorMode, BufferSelect[0]})
        5'b0000_x: UnpackBlue = {Color[ 7: 0], 3'b100};
        5'b0001_x: UnpackBlue = {{11{1'b1}};
        5'b001x_x: UnpackBlue = {Color[ 9: 0], 1'b1};
        5'b0100_0: UnpackBlue = {Color[ 3: 0], 7'b1000000};
```

【０２４０】

【数１４】

```
        5'b0100_1: UnpackBlue = {Color[19:16], 7'b1000000};
        5'b011x_0: UnpackBlue = {Color[ 4: 0], 6'b100000};
        5'b011x_1: UnpackBlue = {Color[20:16], 6'b100000};
        default: UnpackBlue = 'bx;
        endcase
        end
    endfunction
```

＊【０２４１】2.3.9 UnpackDepth

UnpackDepthは、DepthMaskを使用して、ステンシル、オーバレイ、およびWIDデータをマスクアウトする。

【０２４２】

【数１５】

＊40

```
    function [31:0] UnpackDepth;
        input [31:0] Depth;
        input [15:0] DepthMask;
        begin
            UnpackDepth = {(Depth[31:16]&DepthMask),
    Depth[15:0]};
        end
    endfunction
```

【０２４３】2.3.10 UnpackStencil

UnpackStencilは、StencilMaskを使用してデプスデータ をマスクアウトする。

【０２４４】

【数１６】

```
function [7:0] UnpackStencil;
     input [31:0] Extra;
     input [ 7:0] StencilMask;
     begin
          UnpackStencil = Extra[23:16] & StencilMask;
     end
endfunction
```

【０２４５】2.3.11 UnpackWid

UnpackWidは、WidMaskを使用してオーバレイデータをマスクアウトする。

【０２４６】

【数１７】

```
function [7:0] UnpackWid;
     input [31:0] Extra;
     input [ 7:0] WidMask;
     begin
          UnpackWid = Extra[31:24] & WidMask;
     end
endfunction
```

＊【０２４７】2.3.12 SramToPaluData

SramToPaluDataは、SRAMデータの２５６ビットから２つのピクセルについて、３２ビットの色、デプス、およびエクストラ区分を、ならびに、アルファ、赤、緑、青、デプス、ステンシル、およびWIDフィールドを、アンパックする。

【０２４８】

【数１８】

＊

```
function [183:0] SramToPaluData;
     input [255:0] Data;
     input [ 2:0] PixelSize;
     input [ 3:0] PixelAddress;
     input [ 1:0] BufferSelect;
     input [ 3:0] ColorMode;
     input [ 31:0] DepthMask;
     input [ 7:0] StencilMask;
     input [ 7:0] WidMask;

     reg [31:0] Color1, Color0;
     reg [31:0] Depth1, Depth0;
     reg [31:0] Extra1, Extra0;
     begin

          {Color1, Color0} = UnpackColors(Data, PixelSize,
               PixelAddress, BufferSelect);
```

【０２４９】

【数１９】

```
73                                                              74
        {Depth1, Depth0} = UnpackDepths(Data, PixelSize,
             PixelAddress);
        {Extra1, Extra0} = UnpackExtras(Data, PixelSize,
             PixelAddress);

        SramToPaluData = {
             UnpackWid( Extra1, WidMask),
             UnpackStencil(Extra1, StencilMask),
             UnpackDepth( Depth1, DepthMask),
             UnpackAlpha( Color1, ColorMode,
BufferSelect),
             UnpackRed ( Color1, ColorMode, BufferSelect),
             UnpackGreen( Color1, ColorMode,
BufferSelect),
             UnpackBlue ( Color1, ColorMode,
BufferSelect),

             UnpackWid( Extra0, WidMask),
             UnpackStencil(Extra0, StencilMask),
             UnpackDepth( Depth0, DepthMask),
             UnpackAlpha( Color0, ColorMode,
BufferSelect),
             UnpackRed ( Color0, ColorMode, BufferSelect),
             UnpackGreen( Color0, ColorMode,
BufferSelect),
             UnpackBlue ( Color0, ColorMode, BufferSelect)
          };
     end
endfunction
```

【０２５０】2.4 ピクセルALUからSRAMへのデータルーティング

結果データをピクセルALUからSRAMへと送る。Alpha0[9:0], Red0[9:0], Green0[9:0], blue0[9:0], Depth0[31:0], Stencil0[7:0], DT0, ST0, WT0, Alpha1[9:0], Red1[9:0], Green1[9:0], Blue1[9:0], Depth1[31:0], Stencil1[7:0], DT1, ST1,および WT1。SRAMに書込まれるデータの各ビットは、対応の書込イネーブルを有する。

【０２５１】このデータ経路は、ほぼ２つの部分に分割することができる。すなわち、データの３２ビットは「０」側の半分から「１」側の半分に送られなければならない。各半分は、ColorPackユニット、DepthPackユニット、およびExtraPackユニットを有する。ColorPackユニットは、ROP/blendユニットの結果を再フォーマットする。DepthPackユニットは、Depth、StencilおよびAlpha ROP/Blendユニットの結果を再フォーマットする。ExtraPackユニットは、AlphaおよびStencilユニットの結果を再フォーマットする。

【０２５２】2.4.1 PackColor

PackColorは色データの４０ビットを入力として、それをColorModeにしたがって３２ビットワードにパックする。この関数は、すべてのピクセルサイズで使用される。

【０２５３】

【数２０】

75 76

```
function [31:0] PackColor;
      input [3:0] ColorMode
      input [9:0] Alpha, Red, Green Blue;
      begin
            case(ColorMode)
            4'd0:   PackColor   =   {Alpha[9:2],   Red[9:2],   Green[9:2],
Blue[9:2]};
            4'd2:  PackColor  =  {          2'd0,  Red[9:0],  Green[9:0],
Blue[9:0]};
            4'd3:   PackColor   =   {Alpha[1:0],   Red[9:0],   Green[9:0],
Blue[9:0]};
            4'd4:   PackColor   =   {2{Alpha[9:6],   Red[9:6],   Green[9:6],
Blue[9:6]}};
            4'd6: PackColor = {2{Red[9:5], Green[9:4], Blue[9:5]};
            4'd7:   PackColor   =   {2{Alpha[9],   Red[9:5],   Green[9:5],
Blue[9:5]}};
            endcase
      end
endfunction
```

【０２５４】2.4.2 PackDepth

PackDepthは、デプス、ステンシル、およびアルファデータを入力として、それをDepthMaskにしたがって３２ビットワードにパックする。

＊

```
function [31:0] PackDepth;
      input [15:0] DepthMask;
      input [31:0] Depth;
      input [7:0] Stencil;
      input [7:0] Alpha;
      begin
            PackDepth[31:24] = ( DepthMask[15:8] & Depth[31:24]) |
                               (~DepthMask[15:8] & Alpha);
            PackDepth[23:16] = ( DepthMask[ 7:0] & Depth[23:16]) |
                               (~DepthMask[ 7:0] & Stencil);
            PackDepth[15: 0] = Depth[15:0];
      end
endfunction
```

＊ビットワードにパックする。

【０２５５】

【数２１】

【０２５６】2.4.3 PackExtra

PackExtraは、ステンシル、およびアルファデータを入力として、それを３２ビットワードにパックする。

※30

```
function [31:0] PackExtra;
      input [7:0] Stencil,
      input [7:0] Alpha;
      begin
            PackExtra = { Alpha, Stencil, Alpha, Alpha };
      end
endfunction
```

※【０２５７】

【数２２】

【０２５８】2.4.4 PaluToSramData

PaluToSramDataは、ピクセルALUの結果を入力として、それをメモリに書込まれるべき２５６ビットのワードにパックする。PixelSizeおよびPixelAddress入力は、ピクセルフォーマット全体を決定し、一方、ColorMode、AlphaMaskおよびStencilMaskは、データの詳細なフォーマッティングを決定する。

40

【０２５９】

【数２３】

77 78

```
function [255:0] PaluToSramData;
    // Per pixel info
    input  [3:0] PixelAddress;
    input  [9:0] Alpha0, Red0, Green0, Blue0
    input  [9:0] Alpha1, Red1, Green1, Blue1;
    input [31:0] Depth1, Depth0;
    input  [7:0] Stencil1, Stencil0;
    // Register field info
    input  [2:0] PixelSize;
input  [2:0] ColorMode;
    input [31:0] DepthMask;

    reg   [31:0] CP1, CP0, DP1, DP0, EP1, EP0;
    reg  [255:0] Data;
    begin
        CP0 = ColorPack(ColorMode, Alpha0, Red0, Green0, Blue0);
        CP1 = ColorPack(ColorMode, Alpha1, Red1, Green1, Blue1);
        DP0 = DepthPack(DepthMask, Depth0, Stencil0, Alpha0);
        DP1 = DepthPack(DepthMask, Depth1, Stencil1, Alpha1);
        EP0 = { {3{Alpha0}}, Stencil0};
        EP1 = { {3{Alpha1}}, Stencil1};
        casex({PixelSize, PixelAddress})
        7'b000_xxxx: Data = {CP1, CP1, CP1, CP1, CP0, CP0, CP0, CP0};
        7'b001_xxxx: Data = {DP1, CP1, DP1, CP1, DP0, CP0, DP0, CP0};
        7'b010_x0xx: Data = {DP0, DP1, CP1, CP1, CP0, CP0, CP0, CP0};
        7'b010_x1xx: Data = {DP0, DP1, CP1, CP1, CP0, DP0, CP0, CP0};
        7'b011_xxxx: Data = {EP1, DP1, CP1, CP1, EP0, DP0, CP0, CP0};
        endcase

         PaluToSramData = Data;
    end
endfunction
```

【０２６０】2.5 ピクセルALUからSRAMへのマスク生成
ピクセルALU→SRAMデータ経路のための２５６ビット書込マスクを生成する。

【０２６１】2.5.1 WriteEnableMask
WriteEnableMaskは、バイト書込イネーブルビットを２５６ビットワードに拡張する。バイト書込イネーブル＊は、個々の８ビット、１６ビット、および３２ビットピクセルを選択するのに使用される。これらは８ビット、１６ビット、および３２ビットピクセルについてのみ使用されるものである。

【０２６２】

【数２４】

```
function [255:0] WriteEnableMask;
     input [7:0] WriteEnable;
     reg [31:0] Mask1, Mask0;
     begin
          Mask0       =       {       {8{WriteEnable[3]}},
{8{WriteEnable[2]}},
                    {8{WriteEnable[1]}},  {8{WriteEnable[0]}}
};
          Mask1       =       {       {8{WriteEnable[7]}},
{8{WriteEnable[6]}},
                    {8{WriteEnable[5]}},  {8{WriteEnable[4]}}
};
          WriteEnableMask = { {4{Mask1}}, {4{Mask0}} };
     end
endfunction
```

【０２６３】2.5.2 ピクセルアドレスマスク
３２ビットよりも大きいピクセルは、通常、３２ビット区分へと分割される。PixelAddressMaskは、DualPixel、PixelAddress、およびPixelSizeを使用して、２５６ビットワードのどの３２ビット区分に書込がなされるべきかを決定する。

【０２６４】

【数２５】

79 80

```
function [255:0] PixelAddressMask;
    input        DualPixel;
    input  [3:0] PixelAddress;
    input  [2:0] PixelSize;
    reg    [1:0] PixelEnable;
    reg    [7:0] WordEnable;
begin
    casex({PixelSize, DualPixel, PixelAddress})
    8'b000_x_xx00: WordEnable = 8'b0001_0001; // 8,16,32 bit
    8'b000_x_xx01: WordEnable = 8'b0010_0010; // 8,16,32 bit
    8'b000_x_xx10: WordEnable = 8'b0100_0100; // 8,16,32 bit
    8'b000_x_xx11: WordEnable = 8'b1000_1000; // 8,16,32 bit
    8'b001_0_xx00:  WordEnable  =  8'b0000_0011;  //  64  bit
single
    8'b001_0_xx01:  WordEnable  =  8'b0011_0000;  //  64  bit
single
    8'b001_0_xx10:  WordEnable  =  8'b0000_1100;  //  64  bit
single
    8'b001_0_xx11:  WordEnable  =  8'b1100_0000;  //  64  bit
single
    8'b001_1_xx0x: WordEnable = 8'b0011_0011; // 64 bit dual
    8'b001_1_xx1x: WordEnable = 8'b1100_1100; // 64 bit dual

    8'b010_0_x000:  WordEnable  =  8'b1000_0011;  //  96  bit
single
    8'b010_0_x001:  WordEnable  =  8'b0111_0000;  //  96  bit
single
    8'b010_0_x010:  WordEnable  =  8'b1000_1100;  //  96  bit
single
    8'b010_0_x011:  WordEnable  =  8'b0111_0000;  //  96  bit
single
    8'b010_0_x100:  WordEnable  =  8'b0000_0111;  //  96  bit
single
```

【０２６５】

＊　＊【数２６】

```
    8'b010_0_x101:  WordEnable  =  8'b0000_0000;  //  96  bit
single
    8'b010_0_x11x:  WordEnable  =  8'b0000_0000;  //  96  bit
single
    8'b010_1_x00x: WordEnable = 8'b1111_0011; // 96 bit dual
    8'b010_1_x01x: WordEnable = 8'b1111_1100; // 96 bit dual
    8'b010_1_x10x: WordEnable = 8'b0000_0111; // 96 bit dual
    8'b010_1_x11x: WordEnable = 8'b0000_0000; // 96 bit dual
    8'b011_0_xx0x:  WordEnable  =  8'b0000_1111;  //  128  bit
single
    8'b011_0_xx1x:  WordEnable  =  8'b1111_0000;  //  128  bit
single
    8'b011_1_xxxx:  WordEnable  =  8'b1111_1111;  //  128  bit
dual
    endcase

    PixelAddressMask = {
        {32{WordEnable[7]}}, {32{WordEnable[6]}},
        {32{WordEnable[5]}}, {32{WordEnable[4]}},
        {32{WordEnable[3]}}, {32{WordEnable[2]}},
        {32{WordEnable[1]}}, {32{WordEnable[0]}} };
    end
endfunction
```

【０２６６】2.5.3 MaskDepth　　50 MaskDepthは、DepthPackと相似であるが、データ入力が

書込イネーブルに置換されている。

【０２６７】

＊【数２７】

```
function [31:0] MaskDepth;
      input [15:0] DepthMask;
      input        ColorEnable;
      input        DepthEnable;
      input        StencilEnable;
      begin

            MaskDepth[31:24] = { DepthMask[15:8] & {8{DepthEnable}}} |
                               {~DepthMask[15:8] & {8{ColorEnable}}};
            MaskDepth[23:16] = { DepthMask[ 7:0] & {8{DepthEnable}}} |
                               {~DepthMask[ 7:0] & {8{StencilEnable}}};
            MaskDepth[15: 0] = {16{DepthEnable}};

      end
endfunction
```

【０２６８】2.5.4　EnableMask

EnableMaskは、ピクセルの色、デプス、およびステンシルフィールドを個別にイネーブル（またはディセーブル）する。ピクセルALUによって行われるWID、ステンシルおよびデプステストは、どのフィールドに書込がなされるかを決定する。

【０２６９】

【数２８】

```
function [255:0] EnableMask;
    input [3:0] PixelAddress;
    input ColorEnable1,ColorEnable0;
    input DepthEnable1,DepthEnable0;
    input StencilEnable1,StencilEnable0;
    input [2:0] PixelSize;
    input [31:0] DepthMask;
    reg   [31:0] CM1, CM0, DM1, DM0, EM1, EM0;
    begin
        CM0 = {32{ColorEnable0}};
        CM1 = {32{ColorEnable1}};
        DM0 = DepthMask(DepthMask,
                        ColorEnable0, DepthEnable0, StencilEnable0);
        DM1 = DepthMask(DepthMask,
                        ColorEnable1, DepthEnable1, StencilEnable1);
        EM0 = { {24{ColorEnable0}}, {8{StencilEnable0}} };
        EM1 = { {24{ColorEnable1}}, {8{StencilEnable1}} };

        Casex({PixelSize, PixelAddress})
         7'b000-xxxx: EnableMask = {CM1, CM1, CM1, CM1, CM0, CM0, CM0,
CM0};
         7'b001-xxxx: EnableMask = {DM1, CM1, DM1, CM1, DM0, CM0, DM0,
CM0};
         7'b010-x0xx: EnableMask = {DM0, DM1, CM1, CM1, CM0, CM0, CM0,
CM0};
         7'b010-x1xx: EnableMask = {DM0, DM1, CM1, CM1, CM0, DM0, CM0,
CM0};
         7'b011-xxxx: EnableMask = {EM1, DM1, CM1, CM1, EM0, DM0, CM0,
CM0};
        endcase

    end
endfunction
```

【０２７０】2.5.5　SelectPlaneMask

SelectPlaneMaskは、５１２ビットプレーンマスクレジスタから適切な１２８ビット区分を選択する。

【０２７１】

【数２９】

```
function [255:0] SelectPlaneMask;
    input [3:0] PixelAddress;
    input [2:0] PixelSize;
    input [511:0] PlaneMask;
    begin
    casex({PixelSize, PixelAddress})
    7'b00x-x0xx: SelectPlaneMask = PlaneMask[255:0];
    7'b010-x00x: SelectPlaneMask =  PlaneMask[255:0];
    7'b010-x01x:       SelectPlaneMask       =       {PlaneMask[511:384],
PlaneMask[127:0]};
    7'b011-x0xx: SelectPlaneMask = PlaneMask[255:0];
    7'b1xx-x0xx: SelectPlaneMask = PlaneMask[255:0];
    7'bxxx-x1xx: SelectPlaneMask = PlaneMask[511:256];
    endcase
    end
endfunction
```

【０２７２】2.5.6 ピクセルALUからSRAMへのマスク＊スクを生成する。
PaluToSramMaskは、WriteEnableMask、PixelAddressMas【０２７３】
k、EnableMask、およびSelectPlaneMaskの結果に対して【数３０】
ビットごとに論理積をとって、最後のビットごと書込マ＊

```
function [255:0] PaluToSramMask;

    // Per pixel information
    input [7:0] WriteEnable;
    input       DualPixel;
    input [3:0] PixelAddress;
    input       ColorEnable1,ColorEnable0;
    input       DepthEnable1,DepthEnable0;
    input       StencilEnable1,StencilEnable0;

    // Info from register fields
    input   [2:0] PixelSize;
    input  [15:0] AlphaMask;
    input   [7:0] StencilMask;
    input [511:0] PlaneMask;

    begin
        PaluToSramMask
        WriteEnableMask[WriteEnable] &
        PixelAddressMask[DualPixel, PixelAddress, PixelSize] &
        EnableMask[PixelAddress,
            ColorEnable1, ColorEnable0, DepthEnable1, DepthEnable0,
            StencilEnable1, StencilEnable0, PixelSize, DepthMask] &
        SelectPlaneMask[PixelAddress, PixelSize, PlaneMask];
    end
endfunction
```

【０２７４】3.0 ピクセルフォーマット
デュアルピクセル３DRAMチップ１１０は、８ビットから５１２ビットまでの多種多様のピクセルフォーマットをサポートする。８ビットから３２ビットまでのピクセルサイズは、三次元グラフィックスレンダリングオペレーションはサポートしない。２５６ビットおよび５１２ビ※ットのピクセルサイズは、マルチサンプリングされたアンチエイリアシングオペレーションをサポートする。下の表２４は、種々のピクセル速度およびサイズについて、ピークピクセル速度を列挙する。
【０２７５】
【表２４】

表 24　種々のピクセル転送およびサイズに対するピークピクセル速度
（単位：Mピクセル／秒）

| ピクセル当りのビット数 | ML クリア速度 | FL クリア速度 | 2D レンダリング | 3D レンダリング | ディスプレイ |
|---|---|---|---|---|---|
| 8 | 12,800 | 51,200 | 1600 | - | 1600 |
| 16 | 6400 | 25,600 | 800 | - | 800 |
| 32 | 3200 | 12,800 | 400 | - | 400 |
| 64 | 1600 | 6400 | 400 | 267-400 | 400 |
| 128 | 800 | 3200 | 400 | 200-400 | 267 |
| 256 | 400 | 1600 | 200 | 66.7 | 200 |
| 512 | 200 | 800 | 200 | 50 | 200 |

【０２７６】デュアルピクセル３DRAMデバイス１１０の３つの実施例のピクセル容量を、下の表２５に示す。ここで、１Kは１０２４ビットに等しく、１Mは１０２４K＊＊または1，０４８，５７６ビットに等しい。



【０２７７】

【表２５】

表25　ピクセルサイズおよびデバイスサイズによるピクセル容量

| ピクセル当りのビット数 | 40 Mビット | 80 Mビット | 160 Mビット |
|---|---|---|---|
| 8 | 5,242,880 | 10,485,760 | 20,971,520 |
| 16 | 2,621,440 | 5,242,880 | 10,485,760 |
| 32 | 1,310,720 | 2,621,440 | 5,242,880 |
| 64 | 655,360 | 1,310,720 | 2,621,440 |
| 128 | 327,680 | 655,360 | 1,310,720 |
| 256 | 163,840 | 327,680 | 655,360 |
| 512 | 81,920 | 163,840 | 327,680 |

【０２７８】図９８から図１７８は、表２５に示した種々のピクセルフォーマットがデュアルピクセル３DRAMチップ１１０においてどのように処理されるかを示す。

【０２７９】3.１８ビットピクセルフォーマット

図９８から図１０２は、デュアルピクセル３DRAMチップ１１０によって８ビットピクセルをどのように処理することができるかを示す。８ビットピクセルでレンダリングされたグラフィックスについては、表示リフレッシュは１バイト／ピクセルを要し、二次元書込は１バイト／ピクセルを要する。

【０２８０】図９８は、８ビットピクセルからライン１６４へ、およびライン１６４からページ１６２への、考えられるディスプレイマッピングを示す。

【０２８１】図９９は、RPIX、SPIXまたはDPIXオペレーションのための、８ビットピクセルの考えられるキャッシュライン編成を示す。

【０２８２】図１００は、RDATオペレーションを使用する場合に見られる、８ビットピクセルの考えられるキャッシュライン編成を示す。

【０２８３】図１０１は、８ビットピクセルのための考えられるフォーマットを示す。ここで、「I」は８ビットインデックスカラーを表わし、これは、SRAMピクセルバッファ１１８内の２５６ビットエントリをアドレシングするのに使用される。各エントリは、赤につき８ビット、緑につき８ビット、および青につき８ビットを有する。これによりプログラマは、ピクセルあたりわずか８ビットで、可能な１６，７７，２１６（$2^{24}$）色からどの２５６色でも選択することができる。

【０２８４】図１０２は、図１０１に特定されたフォーマットの8ビットピクセルの表示リフレッシュオペレーションが8ピクセルを送信するのに２サイクルのRPIXオペレーションを必要とする様子を示す。

【０２８５】3.２　１６ビットピクセルフォーマット

図１０３から図１１１は、デュアルピクセル３DRAMチップ１１０によって１６ビットピクセルをどのように処理することができるかを示す。二次元ラスタオペレーションは２バイト／ピクセルを要し、二次元ブレンドオペレーションは４バイト／ピクセルを要する。

【０２８６】図１０３は、１６ビットピクセルからライン１６４へ、およびライン１６４からページ１６２への、考えられるディスプレイマッピングを示す。

【０２８７】図１０４は、RPIX、SPIXまたはDPIXオペレーションのための、１６ビットピクセルの考えられるキャッシュライン編成を示す。

【０２８８】図１０５は、RDATオペレーションを使用する場合に見られる、１６ビットピクセルの考えられるキャッシュライン編成を示す。

【０２８９】図１０６は、１６ビットピクセルのための以下のフォーマットを示す：アルファ：４、赤：４、緑：４、青：４。アルファ、赤、緑および青の各成分は、４ビットで表わされる。

【０２９０】図１０７は、図１０６に特定されたフォーマットの４つの１６ビットピクセルを送信するのに、表示リフレッシュオペレーションが２サイクルのRPIXオペレーションを必要とする様子を示す。

【０２９１】図１０８は、１６ビットピクセルのための以下の代替的なフォーマットを示す：赤：５、緑：６、青：５。ここで、赤成分は５ビットで表わされ、緑成分は６ビットで、青成分は５ビットで表わされる。このピクセルフォーマットではアルファ成分は使用されない。

【０２９２】図１０９は、図１０８に特定されたフォーマットの４つの１６ビットピクセルを送信するのに２サイクルのRPIXオペレーションを必要とする、表示リフレッシュオペレーションを示す。

【０２９３】図１１０は、１６ビットピクセルのための以下の代替的なフォーマットを示す：アルファ：１、赤：５、緑：５、青：５。アルファピクセル成分は１ビットで表わされ、赤、緑および青のピクセル成分は各々、５ビットで表わされる。

【０２９４】図１１１は、図１１０に特定されたフォーマットの４つの１６ビットピクセルを送信するのに２サイクルのRPIXオペレーションを必要とする、表示リフレッシュオペレーションを示す。

【０２９５】3.３　３２ビットピクセルフォーマット

図１１２から図１１８は、デュアルピクセル３DRAMチップ１１０によって３２ビットピクセルをどのように処理することができるかを示す。表示リフレッシュは４バイト／ピクセルを要し、二次元書込は４バイト／ピクセルを要する。

【０２９６】図１１２は、３２ビットピクセルからライン１６４へ、およびライン１６４からページ１６２への、考えられるディスプレイマッピングを示す。

【０２９７】図１１３は、RPIX、SPIX、またはDPIXオペレーションのための、３２ビットピクセルの考えられるキャッシュライン編成を示す。

【０２９８】図１１４は、RDATオペレーションを使用する場合に見られる、３２ビットピクセルの考えられるキャッシュライン編成を示す。

【０２９９】図１１５は、３２ビットピクセルのための以下のフォーマットを示す：アルファ：８、赤：８、緑：８、青：８。アルファ、赤、緑および青のピクセル成分は各々、８ビットで表わされる。

【０３００】図１１６は、図１１５に特定されたフォーマットの２つの３２ビットピクセルを送信するのに、表示リフレッシュオペレーションが２サイクルのRPIXオペレーションを必要とする様子を示す。

【０３０１】図１１７は、３２ビットピクセルの以下のフォーマットを示す：赤：１０、緑：１０、青：１０。赤、緑および青のピクセル成分は各々、１０ビットで表わされる。アルファ成分は存在しない。

【０３０２】図１１８は、図１１７に特定されたフォーマットの２つの３２ビットピクセルを送信するのに、表示リフレッシュオペレーションが２サイクルのRPIXオペレーションを必要とする様子を示す。

【０３０３】3.４ ６４ビットピクセルフォーマット

図１１９から図１３７は、デュアルピクセル３DRAM１１０によって６４ビットピクセルをどのように処理することができるかを示す。二次元フィルおよび表示リフレッシュオペレーションについては、２つのピクセルは２サイクル内でアクセスすることができる。

【０３０４】表示リフレッシュは４バイト／ピクセルを要し、二次元レンダリングオペレーションは４バイト／ピクセルを要する。三次元レンダリングオペレーションは、６バイト／ピクセルを要する。

【０３０５】図１１９は、６４ビットピクセルからライン１６４へ、およびライン１６４からページ１６２への、考えられるディスプレイマッピングを示す。

【０３０６】図１２０は、RPIX、SPIX、またはDPIXオペレーションのための、６４ビットピクセルの考えられるキャッシュライン編成を示す。

【０３０７】図１２１は、RDATを使用する場合に見られる、６４ビットピクセルの考えられるキャッシュライン編成を示す。

【０３０８】図１２２は、６４ビットピクセルのための以下のフォーマットを示す：WID：４、アルファ：８、赤：８、緑：８、青：８。アルファ、赤、緑および青のピクセル成分は各々、８ビットで表わされ、ウィンドウIDは４ビットで表わされる。このフォーマットは、三次元のアプリケーションをサポートしない。

【０３０９】図１２３は、図１２２に特定されたフォーマットの２つの６４ビットピクセルを送信するのに、表示リフレッシュオペレーションが２サイクルのRPIXオペレーションを必要とする様子を示す。

【０３１０】図１２４は、６４ビットピクセルのための以下のフォーマットを示す：WID：４、赤：１０、緑：１０、青：１０。アルファ、赤、緑および青のピクセル成分は各々、１０ビットで表わされる。このフォーマットは、三次元のアプリケーションをサポートしない。

【０３１１】図１２５は、図１２４に特定されたフォーマットの２つの６４ビットピクセルを送信するのに、表示リフレッシュオペレーションが２サイクルのRPIXオペレーションを必要とする様子を示す。

【０３１２】図１２６は、６４ビットピクセルのための以下のフォーマットを示す：WID：４、デプス／ステンシル：２８、2*（アルファ：４、赤：４、緑：４、青：４）。このピクセルフォーマットは、各４ビットのアルファ、赤、緑および青のピクセル成分を2組と、２８ビットのデプス／ステンシルフィールドと、４ビットのウィンドウIDフィールドとを含む。

【０３１３】図１２７は、図１２６に特定されたフォーマットの２つの６４ビットピクセルを送信するのに、２サイクルのRPIXオペレーションを必要とする、表示リフレッシュシーケンスを示す。

【０３１４】図１２８は、６４ビットピクセルのための以下のフォーマットを示す：WID：４、オーバレイ：４、デプス／ステンシル：２４、2*（アルファ：４、赤：４、緑：４、青：４）。このピクセルフォーマットは、各々４ビットのアルファ、赤、緑および青のピクセル成分を２組と、２４ビットのデプス／ステンシルフィールドと、４ビットのオーバレイフィールドと、４ビットのウィンドウIDフィールドとを含む。

【０３１５】図１２９は、図１２８に特定されたフォーマットの２つの６４ビットピクセルを送信するのに、表示リフレッシュオペレーションが２サイクルのRPIXオペレーションを必要とする様子を示す。

【０３１６】図１３０は、６４ビットピクセルのための以下のフォーマットを示す：WID：４、デプス／ステンシル：２８、2*（赤：５、緑：６、青：５）。このピクセルフォーマットは、５ビットの赤、６ビットの緑および５ビットの青のピクセル成分を２組と、２８ビットのデプス／ステンシルフィールドと、４ビットのウィンドウIDフィールドとを含む。

【０３１７】図１３１は、図１３０に特定されたフォーマットの２つの６４ビットピクセルを送信するのに、表

示リフレッシュが２サイクルのRPIXオペレーションを必要とする様子を示す。

【０３１８】図１３２は、６４ビットピクセルのための以下のフォーマットを示す：WID：４、オーバレイ：４、デプス／ステンシル：２４、２*（赤：５、緑：６、青：５）。このピクセルフォーマットは、５ビットの赤、６ビットの緑および５ビットの青のピクセル成分フィールドを２組と、２４ビットのデプス／ステンシルフィールドと、４ビットのオーバレイフィールドと、４ビットのウィンドウIDフィールドとを含む。

【０３１９】図１３３は、図１３２に特定されたフォーマットの２つの６４ビットピクセルを送信するのに、表示リフレッシュオペレーションが２サイクルのRPIXオペレーションを必要とする様子を示す。

【０３２０】図１３４は、６４ビットピクセルのための以下のフォーマットを示す：WID：４、デプス／ステンシル：２８、２*（アルファ：１、赤：５、緑：５、青：５）。このフォーマットは、赤、緑および青のピクセル成分につき各々５ビットのフィールドを２組と、アルファピクセル成分のための１ビットフィールドを２組と、２８ビットのデプス／ステンシルフィールドと、４ビットのウィンドウIDフィールドとを含む。

【０３２１】図１３５は、図１３４に特定されたフォーマットの２つの６４ビットピクセルを送信するのに、表示リフレッシュが２サイクルのRPIXオペレーションを必要とする様子を示す。

【０３２２】図１３６は、６４ビットピクセルのための以下のフォーマットを示す：WID：４、オーバレイ：４、デプス／ステンシル：２４、２*（アルファ：１、赤：５、緑：５、青：５）。このフォーマットは、赤、緑および青の各ピクセル成分につき各々5ビットのフィールドを２組と、アルファピクセル成分のための１ビットフィールドを２組と、２４ビットのデプス／ステンシルフィールドと、４ビットのオーバレイフィールドと、４ビットのウィンドウIDフィールドとを含む。

【０３２３】図１３７は、図１３６に特定されたフォーマットを有する２つの６４ビットピクセルを送信するのに、表示リフレッシュが２サイクルのRPIXオペレーションを必要とする様子を示す。

【０３２４】3.5 ９６ビットピクセルフォーマット
図１３８から図１５２は、デュアルピクセル3DRAMチップ１１０によって９６ビットピクセルをどのように処理することができるかを示す。このピクセルフォーマットでは、ピクセルあたりデータの５６ビットが書込まれ、ピクセルあたり４０ビットが表示されて、５ピクセルを８サイクル内にパッキングすることが可能である。表示リフレッシュは6．４バイト／ピクセルを要し、二次元のレンダリングオペレーションは４バイト／ピクセルを要する。

【０３２５】図１３８は、９６ビットピクセルからライン１６４へ、およびライン１６４からページ１６２への、考えられるディスプレイマッピングを示す。

【０３２６】図１３９は、RPIX、SPIX、またはDPIXオペレーションのための、９６ビットピクセルの考えられるキャッシュライン編成を示す。

【０３２７】図１４０は、RDATを使用する場合に見られる、９６ビットピクセルの考えられるキャッシュライン編成を示す。

【０３２８】図１４１は、９６ビットピクセルのための１つのフォーマットを示す：WID：４、デプス／ステンシル：２８、2*（オーバレイ：８、赤：８、緑：８、青：８）。このフォーマットは、オーバレイ、赤、緑および青の各ピクセル成分につき各々８ビットのフィールドを２組と、２８ビットのデプス／ステンシルフィールドと、４ビットのウィンドウIDフィールドとを含む。

【０３２９】図１４２は、表示リフレッシュが、図１４１に特定されたフォーマットを有する２つの９６ビットピクセルを送信するのに３サイクルのRPIXオペレーションを、または、図１４１に特定されたフォーマットを有する１つの９６ビットピクセルを送信するのに２サイクルのRPIXオペレーションを、必要とする様子を示す。

【０３３０】図１４３は、９６ビットピクセルのための以下のフォーマットを示す：WID：４、デプス／ステンシル：２８、2*（アルファ：８、赤：８、緑：８、青：８）。このフォーマットは、アルファ、赤、緑および青の各ピクセル成分につき各々８ビットのフィールドを２組と、２８ビットのデプス／ステンシルフィールドと、４ビットのウィンドウIDフィールドとを含む。

【０３３１】図１４４は、図１４３に特定された９６ビットピクセルフォーマットの表示リフレッシュが、２つのピクセルを送信するのに３サイクルのRPIXオペレーションを、または、１つのピクセルを送信するのに２サイクルのRPIXオペレーションを必要とする様子を示す。

【０３３２】図１４５は、９６ビットピクセルのための以下のフォーマットを示す：WID：４、オーバレイ：４、デプス／ステンシル：２４、2*（アルファ：８、赤：８、緑：８、青：８）。このフォーマットは、アルファ、赤、緑および青の各ピクセル成分につき各々８ビットのフィールドを２組と、４ビットのオーバレイフィールドと、４ビットのウィンドウIDフィールドとを含む。

【０３３３】図１４６は、図１４５に特定された９６ビットピクセルフォーマットの表示リフレッシュが、２つのピクセルを送信するのに３サイクルのRPIXオペレーションを、または、１つのピクセルを送信するのに２サイクルのRPIXオペレーションを必要とする様子を示す。

【０３３４】図１４７は、９６ビットピクセルのための以下のフォーマットを示す：WID：４、デプス／ステンシル：２８、2*（赤：１０、緑：１０、青：１０）。このフォーマットは、赤、緑および青の各ピクセル成分

につき各々１０ビットのフィールドを２組と、２８ビットのデプス／ステンシルフィールドと、４ビットのウィンドウIDフィールドとを含む。

【０３３５】図１４８は、図１４７に特定された９６ビットピクセルフォーマットの表示リフレッシュが、２つのピクセルを送信するのに３サイクルのRPIXオペレーションを、または、１つのピクセルを送信するのに２サイクルのRPIXオペレーションを必要とする様子を示す。

【０３３６】図１４９は、９６ビットピクセルのための以下のフォーマットを示す：WID：４、オーバレイ：４、デプス／ステンシル：２４、2*（赤：１０、緑：１０、青：１０）。このフォーマットは、赤、緑および青の各ピクセル成分につき各々１０ビットのフィールドを２組と、２４ビットのデプス／ステンシルフィールドと、４ビットのオーバレイフィールドと、４ビットのウィンドウIDフィールドとを含む。

【０３３７】図１５０は、図１４９に特定された９６ビットピクセルフォーマットの表示リフレッシュが、２つのピクセルを送信するのに３サイクルのRPIXオペレーションを、または、１つのピクセルを送信するのに２サイクルのRPIXオペレーションを、必要とする様子を示す。

【０３３８】図１５１は、９６ビットピクセルのための以下のフォーマットを示す：WID：４、オーバレイ：４、デプス／ステンシル：２４、4*（アルファ：４、赤：４、緑：４、青：４）。このフォーマットは、アルファ、赤、緑および青の各ピクセル成分につき各々４ビットのフィールドを２組と、２４ビットのデプス／ステンシルフィールドと、４ビットのオーバレイフィールドと、４ビットのウィンドウIDフィールドとを含む。

【０３３９】図１５２は、図１５１に特定された９６ビットピクセルフォーマットの表示リフレッシュが、２つのピクセルを送信するのに３サイクルのRPIXオペレーションを、または、１つのピクセルを送信するのに２サイクルのRPIXオペレーションを必要とする様子を示す。

【０３４０】3.6 １２８ビットピクセルフォーマット
図１５３から図１６１は、デュアルピクセル３DRAMチップ１１０によって１２８ビットピクセルをどのように処理することができるかを示す。このデプスにおいては、ピクセルあたり６４ビットが書込まれ、ピクセルあたり４８ビットが表示されて、４ピクセルを３サイクル内にパッキングすることが可能となる。二次元の性能を高めるために、ピクセルをサイクルあたり２ピクセルの速度で更新することもできる。表示リフレッシュは6バイト／ピクセルを要し、二次元のレンダリングオペレーションは4バイト／ピクセルを要する。１２８ビットピクセルフォーマットはすべて、一定位置の８ビットWIDフィールドを有する。

【０３４１】図１５３は、１２８ビットピクセルからライン１６４へ、およびライン１６４からページ１６２への、考えられるディスプレイマッピングを示す。

【０３４２】図１５４は、RPIX、SPIX、またはDPIXオペレーションのための、１２８ビットピクセルの考えられるキャッシュライン編成を示す。

【０３４３】図１５５は、RDATオペレーションを使用する場合に見られる、１２８ビットピクセルの考えられるキャッシュライン編成を示す。

【０３４４】図１５６は、１２８ビットピクセルのための以下のフォーマットを示す：WID：８、デプス：３２、ステンシル：８、2*（オーバレイ：８、アルファ：８、赤：８、緑：８、青：８）。このフォーマットは、オーバレイ、アルファ、赤、緑および青の各ピクセル成分につき各々８ビットのフィールドを２組と、８ビットのステンシルフィールドと、３２ビットのデプスフィールドと、８ビットのウィンドウIDフィールドとを含む。

【０３４５】図１５７は、図１５６に特定された１２８ビットピクセルフォーマットの表示リフレッシュが、２つのピクセルを送信するのに３サイクルのRPIXオペレーションを必要とする様子を示す。

【０３４６】図１５８は、１２８ビットピクセルのための以下のフォーマットを示す：WID：８、デプス：３２、ステンシル：８、2*（オーバレイ：８、赤：１０、緑：１０、青：１０）。このフォーマットは、赤、緑、および青の各ピクセル成分につき各々１０ビットのフィールドを２組と、オーバレイのための８ビットのフィールドを２組と、８ビットのステンシルフィールドと、３２ビットのデプスフィールドと、８ビットのウィンドウIDフィールドとを含む。

【０３４７】図１５９は、図１５８に特定されたフォーマットを有する１２８ビットピクセルの表示リフレッシュが、２つのピクセルを送信するのに３サイクルのRPIXオペレーションを必要とする様子を示す。

【０３４８】図１６０は、１２８ビットピクセルのための以下のフォーマットを示す：WID：８、ステンシル：８、デプス：３２、2*（アルファ：１０、赤：１０、緑：１０、青：１０）。このフォーマットは、アルファ、赤、緑および青の各ピクセル成分につき各々１０ビットのフィールドを２組と、３２ビットのデプスフィールドと、８ビットのステンシルフィールドと、８ビットのウィンドウIDフィールドとを含む。

【０３４９】図１６１は、図１６０に特定されたフォーマットを有する１２８ビットピクセルの表示リフレッシュが、２つのピクセルを送信するのに３サイクルのRPIXオペレーションを必要とする様子を示す。

【０３５０】3.7 マルチサンプル・ポリゴン・アンチエイリアシング
２５６ビットまたは５１２ビットのフォーマットを有するピクセルは、マルチサンプル・ポリゴン・アンチエイリアシング・レンダリングオペレーションをサポートする。いくつかのポリゴン・アンチエイリアシングのアル

ゴリズムを、それらに関連する利点に焦点を当てて、以下に説明する。

【０３５１】3.7.1 累算バッファ

このアルゴリズムは、複数のエイリアシングされたイメージを累算して、最終のアンチエイリアシングされたイメージを生成する。エイリアシングされた各イメージは、XおよびY方向にオフセットされた異なるサブピクセルでレンダリングされる。エイリアシングされた最終イメージは、累算バッファの現時点における内容に加算される。すべてのエイリアシングされたイメージがレンダリングされかつ累算されると、累算バッファ内のピクセル成分を累算されたイメージの数で除して、最終的なアンチエイリアシングされたイメージが生成される。

【０３５２】累算バッファは、非常に柔軟な技術であって、ポリゴン・アンチエイリアシングに加えて、モーションブラー、フィールドの奥行き、ソフトシャドーに対応することができる。

【０３５３】累算バッファは品質および柔軟性の面で非常に優れているが、この技術はアンチエイリアシングされた最終的なイメージを生成するのに時間がかかりすぎるため、高いフレームレートのアプリケーションには不適切な場合がある。

【０３５４】この技術に関する詳細な説明は、以下の出版物に記載されており、これをここに引用により援用する。ポール・ヘバーリ（Haeberli, Paul）、K.アクレー（K.Akeley）による「累算バッファ：高品質レンダリングのためのハードウェアサポート（"The Accumulation Buffer: Hardware Support for High-Quality Rendering"）」、Computer Graphics、Vol. ２４、No. ４、１９９０年８月、第３０９～３１８頁。

【０３５５】3.7.2 Aバッファ

このアルゴリズムは、各ピクセルについて、ポリゴンフラグメントのソートされたリストを保持する。もし１つの三角形が１つのピクセルを完全にカバーし、かつ完全に不透明である場合、その三角形の背後のフラグメントはすべて捨ててもよい。そうでない場合には、フラグメントはそのリストの適切な場所に挿入される。各フラグメントは、最小でも、色、デプス、ピクセルマスク、および次のフラグメントへのポインタ、の成分を有する。フレーム全体のレンダリングが終わった後に、すべてのピクセルについて最終的な色へのフラグメントリストを決めるのに、付加的な処理が必要である。

【０３５６】Aバッファのアルゴリズムは、透明な三角形を正しくレンダリングするのに非常に優れており、それらの三角形がデプスによってソートされていない場合であっても問題はない。Aバッファのアルゴリズムは通常、三角形が互いに交わる場合には、相当量の付加的な情報が各フラグメントにストアされていない限り、うまく処理することができない。Aバッファルゴリズムの主要な欠点は、ピクセルあたり、無限量の記憶領域および処理を要することである。必要とされるフレームバッファ記憶領域は、１フレーム内のポリゴンの数にほぼ比例する。

【０３５７】この技術の詳細な説明は、以下の出版物に記載されており、これをここに引用により援用する。ローレン・カーペンター（Carpenter, Loren）による「Aバッファ、アンチエイリアス隠面の方法（"The A-buffer, an Anti-aliased HiddenSurface Method"）」、Computer Graphics、Vol. １８、No. ３、１９８４年７月、第１０３～１０８頁。

【０３５８】3.7.3 マルチサンプル

マルチサンプルアンチエイリアシングは、ピクセルあたりいくつかのサンプルをストアする。各サンプルは、そのピクセル内またはそのピクセルの近辺の異なる場所に位置決めされる。サンプルは、色情報をストアするか、または、色、デプスおよびステンシルの情報をストアする。ピクセルをレンダリングするとき、三角形の内部に存在するサンプルが算出されて、フレームバッファ内のサンプルとマージされる。フレーム全体がレンダリングされた後に、サンプルのすべての色の重み付平均が、表示装置に送られる。

【０３５９】図１６２は、２つのピクセルおよび、各ピクセル内の各サンプルの場所を示す。現時点の三角形の内部のサンプルは中黒の丸で示し、その三角形の外部のサンプルは中空の丸で示す。

【０３６０】マルチサンプルアンチエイリアシングは、フレームあたりレンダリングされる三角形の数にかかわらず、ピクセルあたり一定量の記憶領域を必要とする。この技術は、OpenGLまたはDirectXに対するAPIの変更をほとんど必要とせず、必要なのは、アンチエイリアシング機能をイネーブルまたはディセーブルするための何らかの方法のみである。

【０３６１】この技術の詳細な説明は以下の出版物に記載されており、これをここに引用により援用する。カート・アクレー（Akeley, Kurt）による「リアリティエンジン・グラフィックス（"RealityEngine Graphics"）」、Computer Graphics、１９９３年８月、第１０９～１１６頁。

【０３６２】3.7.3.1 サンプルあたり色のみ

この場合、各ピクセルはWID、ステンシル、デプス、バックカラー、フロントカラーのための記憶領域を有し、各サンプルは色のための記憶領域を有する。レンダリングコントローラ１０２は、共通のピクセル色、共通のピクセルデプス、および、そのサンプルが三角形の内部にあるか否かを示すサンプルあたり１ビットを送る。三角形の辺は、正しくアンチエイリアシングされる。ただし、三角形の交点は、ピクセルがデプス値を１つしかストアしていないため、エイリアシングされるであろう。これを下に、verilogコードで説明する。

【０３６３】

【数３１】

```
if WID test passes {
    update stencil
    if stencil and depth tests pass {
        colorsum = 0
        for each sample {
            if sample is inside triangle
                merge source color with sample's color
            colorsum. += sample's color
        }
        back color = colorsum / # of samples
        overwrite depth
    }
}
```

【０３６４】3.7.3.2 サンプルあたり色およびデプス
この場合、各ピクセルはWID、バックカラー、およびフロントカラーのための記憶領域を有し、各サンプルは色、デプス、およびステンシルのための記憶領域を有する。レンダリングコントローラ１０２は、共通のピクセル色、サンプルあたり１つの異なるデプス値、および、＊そのサンプルが三角形の内部にあるか否かを示すサンプルあたり１ビットを送る。三角形の辺および交点の両方が、正しくアンチエイリアシングされるであろう。これを下に、verilogコードで説明する。

【０３６５】

【数３２】

```
if WID test passes {
    colorsum = 0
    for each sample {
        if sample is inside triangle {
            update sample stencil
            if sample's stencil and depth tests pass {
                merge source color with sample color
                overwrite sample depth
            }
        }
        colorsum += sample color
    }
    back color = colorsum / # of samples
}
```

【０３６６】3.7.4 サンプルあたり色およびデプスの速度改良
デプス値は通常、次のように、XおよびYの１次関数として計算される。

【０３６７】

【数３３】

$$Depth(X, Y) = \frac{dDepth}{dX}X + \frac{dDepth}{dY}Y + Depth(0,0)$$

※【０３６８】この等式は下のように書き換えることができるが、この方がより有益である。

【０３６９】

【数３４】

※

$$Depth(X + \Delta X, Y + \Delta Y) = Depth(X, Y) + \frac{dDepth}{dX}\Delta X + \frac{dDepth}{dY}\Delta Y$$

【０３７０】通常、ピクセル内のサンプル位置は、図１６３に示すように、ピクセル内の中央、隅部、またはある固定された基準点から、XおよびY方向に同じだけオフセットされている。

【０３７１】もし、ピクセルの中央（または他の何らかの基準点）におけるデプスがわかっていれば、サンプルnのデプスは、次の式で計算することができる。

【０３７２】

【数３５】

97 98

$$Depth(X + \Delta X_n, Y + \Delta Y_n) = Depth(X, Y) + \frac{dDepth}{dX}\Delta X_n + \frac{dDepth}{dY}\Delta Y_n$$

【０３７３】この計算は、２つのステップで行なうことができる。

【０３７４】

【数３６】

$$\Delta Depth_n = \frac{dDepth}{dX}\Delta X_n + \frac{dDepth}{dY}\Delta Y_n$$

$$Depth(X + \Delta X_n, Y + \Delta Y_n) = Depth(X, Y) + \Delta Depth_n$$

ここで、$\frac{dDepth}{dX}$ および $\frac{dDepth}{dY}$ の項は、同じ三角形の中のすべてのピクセルについて同じである。したがって、$\Delta Depth_n$ の項もまた、同じ三角形内のすべてのピクセルについて同じとなる。

【０３７５】１つの三角形につき、各サンプルの（$Depth_n$ 項をすべて一度に計算して、それらをレジスタ書込としてデュアルピクセル３DRAMのピクセルALUへと、その三角形内の最初のピクセルをレンダリングする前に、送信することが可能である。その後、その三角形内の各ピクセルについて、ピクセルの中央（または何らかの他の基準点）における色およびデプス値のみが送信される。各サンプルにおけるデプス値は、以下の式を用いて計算される。

【０３７６】

【数３７】

$$Depth(X+\Delta X_n, Y + \Delta Y_n) = Depth(X, Y) + \Delta Depth_n$$

【０３７７】もし各サンプルがデュアルピクセル３DRAMのピクセルALU内に専用のデプス加算器、デプス比較器、および色ブレンドユニットを有する場合には、アンチエイリアシングされたピクセル全体を１つのオペレーションでレンダリングすることが可能であろう。デュアルピクセル３DRAMチップ１１０は、このような能力を有する。

【０３７８】3.8 ２５６ビットピクセルフォーマット（４×マルチサンプル）

図１６４から図１６９は、デュアルピクセル３DRAMチップ１１０によって２５６ビットピクセルをどのように処理することができるかを示す。１２８ビットピクセルに加えて、チップ１１０は、ピクセルあたり４つのサブサンプルをストアすることができる。ピクセルを書込むために、チップ１１０は色およびデプス値、ならびに、４ビットのサンプルマスクを送る。ピクセルALUがサイクルあたり２つのサンプルを処理することができる場合、チップ１１０は２サイクルで各ピクセルを更新することが可能である。ピクセルが更新される間、４つのサンプルのすべての色値が累算されて、AまたはB色バッファに書込まれる。

【０３７９】マルチサンプルアンチエイリアシングは、Aバッファをベースとするアンチエイリアシングよりも実装が容易である。マルチサンプルは、ピクセルを更新するのに、ピクセルあたり大量ではあるが有限量の記憶領域と、一定量の時間とを要するが、フレームのレンダリングとそのフレームの表示との間に、フラグメント決定処理ステージを必要としない。三角形の速度が１秒あたり数百万個であるとき、ほとんどのピクセルは部分的にカバーされており、Aバッファのフィル速度は、フラグメントの記憶領域の要求が飛躍的に増大するのにつれて低速化するであろう。

【０３８０】表示リフレッシュは８バイト／ピクセルを要し、二次元のレンダリングは４バイト／ピクセルを、三次元のレンダリングは９バイト／ピクセルを要する。

【０３８１】２５６ビットピクセルフォーマットはすべて、一定位置の８ビットWIDフィールドを有する。すべてのレンダリングフォーマットは、２サイクルのSPIXオペレーションまたは２サイクルのDPIXオペレーションを要する。表示リフレッシュは、１つのピクセルを送信するのに、２サイクルのRPIXオペレーションを必要とする。

【０３８２】図１６４は、２５６ビットピクセルからライン１６４へ、および、ライン１６４からページ１６２への、考えられるディスプレイマッピングを示す。

【０３８３】図１６５は、RPIX、SPIX、およびDPIXオペレーションのための、２５６ビットピクセルの考えられるキャッシュライン編成を示す。

【０３８４】図１６６は、RDATオペレーションを使用する場合に見られる、２５６ビットピクセルの考えられるキャッシュライン編成を示す。

【０３８５】図１６７は、２５６ビットピクセルのための以下のフォーマットを示す：WID：８、ステンシル：８、デプス：３２、2*（オーバレイ：８、アルファ：８、赤：８、緑：８、青：８）、4*（アルファ：８、赤：８、緑：８、青：８）。このフォーマットは、ウィンドウIDおよびステンシルにつき各８ビットのフィールドと、デプスのための３２ビットのフィールドと、オーバレイ、アルファ、赤、緑および青の各ピクセル成分につき各々８ビットのフィールドを２組と、さらに、アルファ、赤、緑および青の各８ビットの成分を含む４つの色サンプルとを含む。

【０３８６】図１６８は、図１６７に特定された２５６ビットピクセルフォーマットの表示リフレッシュが、１つのピクセルを送信するのに２サイクルのRPIXオペレーションを必要とする様子を示す。

【０３８７】図１６９は、２５６ビットピクセルフォーマットのための、SRAM読出／書込フォーマットを示す。

【０３８８】3.9 ５１２ビットピクセルフォーマット（６×マルチサンプル）

図１７０から図１７８は、デュアルピクセル３DRAMチップ１１０によって５１２ビットピクセルをどのように処理することができるかを示す。このピクセルフォーマットは、６×マルチサンプリングオペレーションをサポートする。このフォーマットにおいて、ピクセルあたり６つのサブサンプルが、１２８ビットピクセルに加えてストアされる。ピクセルを書込むために、共通の色値が最初に送信され、これに６つのデプス値が続く。もしピクセルALUが１サイクルあたり２つのサンプルを処理することができれば、各ピクセルは８サイクルで更新することが可能である。ピクセルが更新されている間に、６つのサンプルのすべての色値が累算されて、AまたはB色バッファに書込まれる。

【０３８９】マルチサンプリングによるアンチエイリアシングは、Aバッファをベースとするアンチエイリアシングよりも実装が容易である。マルチサンプリングは、ピクセルを更新するのに、ピクセルあたり大量ではあるが有限量の記憶領域、および、一定量の時間を必要とするが、フレームのレンダリングとそのフレームの表示との間に、フラグメント決定ステージを必要とはしない。このマルチサンプリングの技術は、互いに貫通する面をアンチエイリアシングする。Aバッファを強化して互いに貫通する面をアンチエイリアシングするのは費用が高くつく。三角形の速度が１秒あたり数百万個である場合、ほとんどのピクセルは部分的にカバーされており、Aバッファのフィル速度は、フラグメント記憶領域の要件が大いに増大する一方で、低速化することになる。

【０３９０】表示リフレッシュは８バイト／ピクセルを要し、二次元のレンダリングオペレーションは４バイト／ピクセルを、三次元のレンダリングオペレーションは３２バイト／ピクセルを要する。

【０３９１】５１２ビットピクセルフォーマットはすべて、一定位置の８ビットWIDフィールドを有する。

【０３９２】図１７０は、５１２ビットピクセルからライン１６４へ、および、ライン１６４からページ１６２への、考えられるディスプレイマッピングを示す。

【０３９３】図１７１は、RPIX、SPIXまたはDPIXオペレーションのための、５１２ビットピクセルの考えられるキャッシュライン編成を示す。

【０３９４】図１７２は、RDATオペレーションを使用する場合に見られる、５１２ビットピクセルの考えられるキャッシュライン編成を示す。

【０３９５】図１７３は、５１２ビットピクセルのための１つのフォーマットを示す：WID：８、２*（オーバレイ：８、アルファ：８、赤：８、緑：８、青：８）、６*（アルファ：８、赤：８、緑：８、青：８、ステンシル／デプス：３２）。このフォーマットは、各サンプルがアルファ、赤、緑および青の各ピクセル成分につき各々８ビットのフィールドおよびデプス／ステンシルのための３２ビットのフィールドを有する６つのピクセルサンプルと、オーバレイ、アルファ、赤、緑および青の各ピクセル成分につき各々８ビットのフィールドを２組と、８ビットのウィンドウIDとを含む。

【０３９６】図１７４は、図１７３に特定された５１２ビットピクセルフォーマットの表示リフレッシュが、１つのピクセルを送信するのに２サイクルのRPIXオペレーションを必要とする様子を示す。

【０３９７】図１７５は、図１７３に特定されたフォーマットの５１２ビットピクセルのための、SRAMピクセルバッファ１１８への読出フォーマットおよびSRAMピクセルバッファ１１８からの書込フォーマットを示す。

【０３９８】図１７６は、５１２ビットピクセルのための以下のフォーマットを示す：WID：８、２*（オーバレイ：８、赤：１０、緑：１０、青：１０）、６*（赤：１０、緑：１０、青：１０、ステンシル／デプス：３２）。このフォーマットは、各サンプルが赤、緑および青の各ピクセル成分につき各々１０ビットのフィールドおよびデプス／ステンシルのための３２ビットのフィールドを有する６つのピクセルサンプルと、赤、緑および青の各ピクセル成分につき各々１０ビットのフィールドを２組と、オーバレイのための８ビットのフィールドを２組と、８ビットのウィンドウIDとを含む。

【０３９９】図１７７は、図１７６に特定された５１２ビットピクセルフォーマットの表示リフレッシュが、１つのピクセルを送信するのに２サイクルのRPIXオペレーションを必要とする様子を示す。

【０４００】図１７８は、図１７６に特定されたフォーマットの５１２ビットピクセルのための、SRAMピクセルバッファ１１８への読出フォーマットおよびSRAMピクセルバッファ１１８からの書込フォーマットを示す。

【０４０１】4.0 双方向I/O

高速かつ同時の双方向送受信の実行可能ないくつかの実装例が、最近提示されてきている。この技術は、２つのデバイスをギガビット速度で接続する単一の配線を介して、データを双方向で送信することを可能にする。このセクションでは、このI/O技術を性能の向上またはコストの低減のためにデュアルピクセル３DRAMデバイスにどのように適用することができるかを示す。開示される高速かつ同時の双方向送受信の詳細な説明は、以下の出版物に提示されており、これをここに引用により援用する。M.ヘイコック（Haycock, M.）、R.ムーニー（Mooney, R.）による「２．５Gb/s双方向送受信技術（"A 2.5

Gb/s Bidirectional Signaling Technology") 」、Hot Interconnects Symposium V、１９９７年８月、第１４９～１５６頁。

【０４０２】図１７９は、データピンを時分割するのに代えて、制御／アドレスピンを介して表示リフレッシュデータを送ることによって、性能を向上させる方式を図示する。この表示リフレッシュ経路は、表示リフレッシュ情報をフェッチするために制御およびアドレス情報を生成するのに、ステートマシンを必要とする。データピン上のトラフィックは通常、単一方向である。

【０４０３】図１８０は、ピクセルALUをレンダリングコントローラへと移動させることを可能にする方式を図示する。これらデータピンは、ソースおよび結果ピクセルデータの同時送信を可能にする。制御／アドレス情報および表示リフレッシュデータは、ピンの同じ組を共有する。

【図面の簡単な説明】

【図１】１組のデュアルピクセル３DRAMチップと、レンダリングコントローラと、レンダリングバスとアドレスおよび制御バスとの対と、ビデオ出力回路とを含む、ビデオディスプレイフレームバッファとも称されるグラフィックスサブシステムを示すブロック図である。

【図２】DRAMアレイ、SRAMピクセルバッファ、２つのピクセル算術論理演算装置（ALU）およびグローバルバスを含むデュアルピクセル３DRAMチップの一実施例を示す機能ブロック図である。

【図３】RAMBUS™またはSyncLink入力／出力インタフェース仕様のいずれかで動作するよう構成される、図２に示される要素を含むデュアルピクセル３DRAMチップの一実施例のためのダイサイズフロアプランの図である。

【図４】いくつかの処理要素の回路のある部分が共有される、２組の処理要素を含むピクセルALUの一実施例のブロック図である。

【図５】２つの別個の組の処理要素を含むピクセルALUの一実施例のブロック図である。

【図６】ピクセルALUの一実施例のための、４つのラスタオペレーション（ROP）／ブレンドユニット（ピクセルのアルファ、赤、緑および青成分の各々を処理するために１つずつ）と、デプスユニットと、ステンシルユニットと、ウインドウ識別（ID）ユニットとを含む処理要素の完全な組と、これらの処理要素に入力され、これらの処理要素から出力される情報の種類とを示す図である。

【図７】１つのROP／ブレンドユニットの一実施例を示すブロック図である。

【図８】１つのラスタオペレーション（ROP）ユニットの一実施例を示すブロック図である。

【図９】１つの８ビットブレンドユニットの一実施例を示すブロック図である。

【図１０】１組の８ビットブレンドユニット計算を示す図である。

【図１１】ブレンドユニットにおいて用いられるドット分散型組織的ディザアルゴリズムを示す図である。

【図１２】１０ビットブレンドユニットの一実施例を示すブロック図である。

【図１３】１組の１０ビットブレンドユニット計算を示す図である。

【図１４】一実施例のためのROP／ブレンドユニットへの入力マルチプレクサの詳細を示す図である。

【図１５】デプス比較ユニットの一実施例を示すブロック図である。

【図１６】ステンシル比較ユニットの一実施例を示すブロック図である。

【図１７】ステンシルデータ経路の一実施例を示すブロック図である。

【図１８】ウインドウID比較ユニットの一実施例を示すブロック図である。

【図１９】SRAMピクセルバッファの一実施例を示すブロック図である。

【図２０】SRAMピクセルバッファの第２の実施例を示すブロック図である。

【図２１】高レベルでのデュアルピクセル３DRAMチップのメモリ構成を示す図である。

【図２２】デュアルピクセル３DRAMチップのメモリ構成を示すより詳細なブロック図である。

【図２３】SRAMピクセルバッファに関連してDRAMバンクメモリ構成の一実施例を示すブロック図である。

【図２４】アドレスおよび制御ポートを示すブロック図である。

【図２５】アドレスおよび制御（RQ）ピンを介するデュアルピクセル３DRAMプロトコル構造を示す図である。

【図２６】アドレスおよび制御ピンを介するバンクオペレーションプロトコルを示す図である。

【図２７】ページプリチャージオペレーションを示すタイミングダイヤグラムの図である。

【図２８】ページアクセスオペレーションを示すタイミングダイヤグラムの図である。

【図２９】ページ変更オペレーションを示すタイミングダイヤグラムの図である。

【図３０】アドレスおよび制御（RQ）ピンを介するアイドルコマンドを示す図である。

【図３１】アドレスおよび制御（RQ）ピンを介するRead Cache Line（RL）コマンド、Write Cache Line（WL）コマンド、Masked Write Cache Line（ML）コマンドおよびChange Cache Line（CL）コマンドを示す図である。

【図３２】アドレスおよび制御（RQ）ピンを介するFlash Masked Write Cache Lineコマンドを示す図である。

【図３３】Read Cache Line（RL）オペレーションを示すタイミングダイヤグラムの図である。

【図３４】Write Cache Line (WL) オペレーションを示すタイミングダイヤグラムの図である。
【図３５】Masked Write Cache Line (ML) オペレーションを示すタイミングダイヤグラムの図である。
【図３６】Flash Masked Write Cache Line (FL) オペレーションを示すタイミングダイヤグラムの図である。
【図３７】Change Cache Line (CL) オペレーションを示すタイミングダイヤグラムの図である。
【図３８】アドレスおよび制御 (RQ) ピンを介するアイドルコマンドを示す図である。
【図３９】アドレスおよび制御 (RQ) ピンを介するRead Data (RDAT) コマンド、Write Data (WDAT) コマンドおよびBroadcast Data (BDAT) コマンドを示す図である。
【図４０】アドレスおよび制御ピンを介するRead Register (RREG) コマンド、Write Register (WREG) コマンドおよびBroadcast Register (BREG) コマンドを示す図である。
【図４１】アドレスおよび制御ピンを介するRead Pixel (RPIX) コマンド、Single Pixel (SPIX) コマンドおよびDual Pixel (DPIX) コマンドを示す図である。
【図４２】データ (DQ) ピンを介するグラフィックスデータ転送を示す図である。
【図４３】２サイクルRDATオペレーションとその後の３サイクルRPIXオペレーションとを示すタイミングダイヤグラムの図である。
【図４４】WDAT、BDAT、WREGおよびDREGオペレーションのためのタイミングダイヤグラムの図である。
【図４５】SPIXおよびDPIXオペレーションのためのタイミングダイヤグラムの図である。
【図４６】３サイクルDPIX転送オペレーションを示すタイミングダイヤグラムの図である。
【図４７】複合２サイクル読出および２サイクル書込オペレーションを示すタイミングダイヤグラムの図である。
【図４８】複合２サイクル読出および３サイクル書込オペレーションを示すタイミングダイヤグラムの図である。
【図４９】複合３サイクル読出および２サイクル書込オペレーションを示すタイミングダイヤグラムの図である。
【図５０】複合３サイクル読出および３サイクル書込オペレーションを示すタイミングダイヤグラムの図である。
【図５１】４つの２サイクル読出オペレーションを示すタイミングダイヤグラムの図である。
【図５２】８つの２サイクルDPIXオペレーションを示すタイミングダイヤグラムの図である。
【図５３】図５２の８つの２サイクルDPIXオペレーションを続けて示す図である。
【図５４】いくつかのレジスタのためのデータフォーマットを示す図である。
【図５５】PixelConfigレジスタのためのデータフォーマットを示す図である。
【図５６】StencilDepthConfigレジスタのためのデータフォーマットを示す図である。
【図５７】ColorOp[0]レジスタのためのデータフォーマットを示す図である。
【図５８】ColorOp[1]レジスタのためのデータフォーマットを示す図である。
【図５９】ConstantColorレジスタのためのデータフォーマットを示す図である。
【図６０】DisplayConfigレジスタのためのデータフォーマットを示す図である。
【図６１】WREGオペレーションからByteMaskおよびMLオペレーションによりどのように高速フィルを行なうかを示すタイミングダイヤグラムの図である。
【図６２】FLオペレーションによってどのように非常に高速のフィルを行なうかを示すタイミングダイヤグラムの図である。
【図６３】入力データフォーマッタを示すブロック図である。
【図６４】入力データフォーマッタの入力ルーティング層を示す図である。
【図６５】入力データフォーマットを示す図である。
【図６６】色およびデプスアキュムレータを示すブロック図である。
【図６７】ROP／ブレンドユニットのための色成分フォーマッティングを示す図である。
【図６８】出力データフォーマッタを示すブロック図である。
【図６９】RDAT、RPIXオペレーションの出力データルーティングを示す図である。
【図７０】出力データルーティングを示す図である。
【図７１】RPIXオペレーションの出力データルーティングを示す図である。
【図７２】出力データルーティングを示す図である。
【図７３】出力データルーティングを示す図である。
【図７４】出力データルーティングを示す図である。
【図７５】９６ビット／ピクセルの出力データルーティングを示す図である。
【図７６】出力データルーティングを示す図である。
【図７７】出力データルーティングを示す図である。
【図７８】出力データルーティングを示す図である。
【図７９】出力データルーティングを示す図である。
【図８０】出力データルーティングを示す図である。
【図８１】出力データルーティングを示す図である。
【図８２】出力データルーティングを示す図である。
【図８３】出力データルーティングを示す図である。
【図８４】SRAM－ピクセルALUフォーマッタを示すブロ

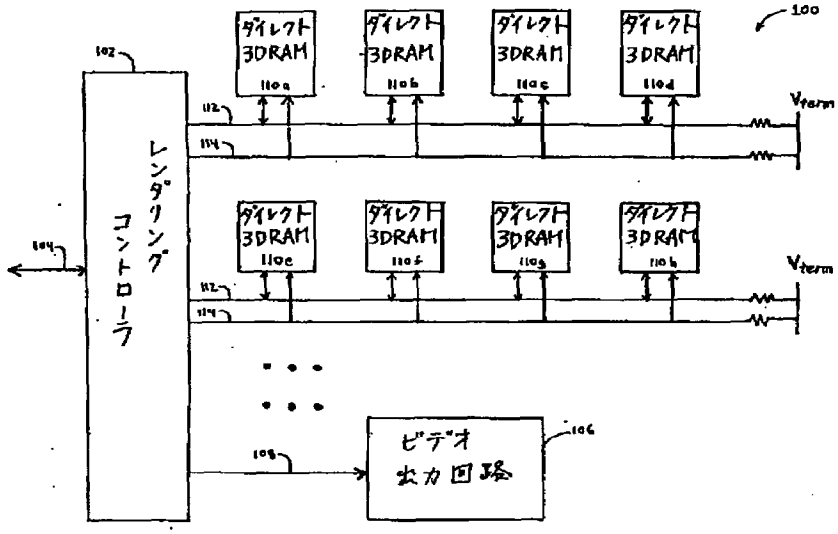
ック図である。
【図８５】８、１６、３２ビット／ピクセルのSRAMピクセルバッファ編成を示す図である。
【図８６】６４ビット／ピクセルのSRAMピクセルバッファ編成を示す図である。
【図８７】９６ビット／ピクセルのSRAMピクセルバッファ編成を示す図である。
【図８８】１２８ビット／ピクセルのSRAMピクセルバッファ編成を示す図である。
【図８９】アンパック関数を示す図である。
【図９０】UnpackDepth関数を示す図である。
【図９１】UnpackStencil関数を示す図である。
【図９２】UnpackWid関数を示す図である。
【図９３】ピクセルALU－SRAMフォーマッタを示すブロック図である。
【図９４】PackColor関数を示す図である。
【図９５】PackDepth関数を示す図である。
【図９６】PackExtra関数を示す図である。
【図９７】ピクセルALU－SRAMマスク生成を示す図である。
【図９８】８ビット／ピクセルディスプレイマッピングを示す図である。
【図９９】RPIX、SPIXおよびDPIXオペレーションのための８ビット／ピクセルキャッシュライン編成を示す図である。
【図１００】RDATオペレーションを用いる８ビット／ピクセルキャッシュライン編成を示す図である。
【図１０１】８ビット／ピクセルフォーマットを示す図である。
【図１０２】８ビット／ピクセル表示リフレッシュを示す図である。
【図１０３】１６ビット／ピクセルディスプレイマッピングを示す図である。
【図１０４】RPIX、SPIXおよびDPIXオペレーションのための１６ビット／ピクセルキャッシュライン編成を示す図である。
【図１０５】RDATオペレーションを用いる１６ビット／ピクセルキャッシュライン編成を示す図である。
【図１０６】4:4:4:4の１６ビット／ピクセルフォーマットを示す図である。
【図１０７】4:4:4:4の１６ビット／ピクセル表示リフレッシュを示す図である。
【図１０８】5:6:5の１６ビット／ピクセルフォーマットを示す図である。
【図１０９】5:6:5の１６ビット／ピクセル表示リフレッシュを示す図である。
【図１１０】1:5:5:5の１６ビット／ピクセルフォーマットを示す図である。
【図１１１】1:5:5:5の１６ビット／ピクセル表示リフレッシュを示す図である。
【図１１２】３２ビット／ピクセルディスプレイマッピングを示す図である。
【図１１３】RPIX、SPIXおよびDPIXオペレーションのための３２ビット／ピクセルキャッシュライン編成を示す図である。
【図１１４】RDATオペレーションを用いる３２ビット／ピクセルキャッシュライン編成を示す図である。
【図１１５】8:8:8:8の３２ビット／ピクセルフォーマットを示す図である。
【図１１６】8:8:8:8の３２ビット／ピクセル表示リフレッシュを示す図である。
【図１１７】10:10:10:10の３２ビット／ピクセルフォーマットを示す図である。
【図１１８】2:10:10:10の３２ビット／ピクセル表示リフレッシュを示す図である。
【図１１９】６４ビット／ピクセルディスプレイマッピングを示す図である。
【図１２０】RPIX、SPIXおよびDPIXオペレーションのための６４ビット／ピクセルキャッシュライン編成を示す図である。
【図１２１】RDATオペレーションを用いる６４ビット／ピクセルキャッシュライン編成を示す図である。
【図１２２】4:8:8:8:8の６４ビット／ピクセルフォーマットを示す図である。
【図１２３】4:8:8:8:8の６４ビット／ピクセル表示リフレッシュを示す図である。
【図１２４】2:10:10:10の６４ビット／ピクセルフォーマットを示す図である。
【図１２５】4:10:10:10の６４ビット／ピクセル表示リフレッシュを示す図である。
【図１２６】4:28:2* (4:4:4:4) の６４ビット／ピクセルフォーマットを示す図である。
【図１２７】4:28:2* (4:4:4:4) の６４ビット／ピクセル表示リフレッシュを示す図である。
【図１２８】4:4:24:2* (4:4:4:4) の６４ビット／ピクセルフォーマットを示す図である。
【図１２９】4:4:24:2* (4:4:4:4) の６４ビット／ピクセル表示リフレッシュを示す図である。
【図１３０】4:28:2* (5:6:5) の６４ビット／ピクセルフォーマットを示す図である。
【図１３１】4:28:2* (5:6:5) の６４ビット／ピクセル表示リフレッシュを示す図である。
【図１３２】4:4:24:2* (5:6:5) の６４ビット／ピクセルフォーマットを示す図である。
【図１３３】4:4:24:2* (5:6:5) の６４ビット／ピクセル表示リフレッシュを示す図である。
【図１３４】4:28:2* (1:5:5:5) の６４ビット／ピクセルフォーマットを示す図である。
【図１３５】4:28:2* (1:5:5:5) の６４ビット／ピクセル表示リフレッシュを示す図である。

【図１３６】4:4:24:2* (1:5:5:5) の６４ビット／ピクセルフォーマットを示す図である。
【図１３７】4:4:24:2* (1:5:5:5) の６４ビット／ピクセル表示リフレッシュを示す図である。
【図１３８】９６ビット／ピクセルディスプレイマッピングを示す図である。
【図１３９】RPIX、SPIXおよびDPIXオペレーションのための９６ビット／ピクセルキャッシュライン編成を示す図である。
【図１４０】RDATオペレーションを用いる９６ビット／ピクセルキャッシュライン編成を示す図である。
【図１４１】4:28:2* (8:8:8:8) の９６ビット／ピクセルフォーマットを示す図である。
【図１４２】4:28:2* (8:8:8:8) の９６ビット／ピクセル表示リフレッシュを示す図である。
【図１４３】4:28:2* (8:8:8:8) の９６ビット／ピクセルフォーマットを示す図である。
【図１４４】4:28:2* (8:8:8:8) の９６ビット／ピクセル表示リフレッシュを示す図である。
【図１４５】4:4:24:2* (8:8:8:8) の９６ビット／ピクセルフォーマットを示す図である。
【図１４６】4:4:24:2* (8:8:8:8) の９６ビット／ピクセル表示リフレッシュを示す図である。
【図１４７】4:28:2* (2:10:10:10) の９６ビット／ピクセルフォーマットを示す図である。
【図１４８】4:28:2* (2:10:10:10) の９６ビット／ピクセル表示リフレッシュを示す図である。
【図１４９】4:4:24:2* (10:10:10) の９６ビット／ピクセルフォーマットを示す図である。
【図１５０】4:4:24:2* (10:10:10) の９６ビット／ピクセル表示リフレッシュを示す図である。
【図１５１】4:4:24:4* (4:4:4:4) の９６ビット／ピクセルフォーマットを示す図である。
【図１５２】4:4:24:4* (4:4:4:4) の９６ビット／ピクセル表示リフレッシュを示す図である。
【図１５３】１２８ビット／ピクセルディスプレイマッピングを示す図である。
【図１５４】RPIX、SPIXおよびDPIXオペレーションのための１２８ビット／ピクセルキャッシュライン編成を示す図である。
【図１５５】RDATオペレーションを用いる１２８ビット／ピクセルキャッシュライン編成を示す図である。
【図１５６】8:32:8:2* (8:8:8:8:8) の１２８ビット／ピクセルフォーマットを示す図である。
【図１５７】8:32:8:2* (8:8:8:8:8) の１２８ビット／ピクセル表示リフレッシュを示す図である。
【図１５８】8:32:8:2* (8:2:10:10:10) の１２８ビット／ピクセルフォーマットを示す図である。
【図１５９】8:32:8:2* (8:10:10:10) の１２８ビット／ピクセル表示リフレッシュを示す図である。
【図１６０】8:8:32:2* (10:10:10:10) の１２８ビット／ピクセルフォーマットを示す図である。
【図１６１】8:8:32:2* (10:10:10:10) の１２８ビット／ピクセル表示リフレッシュを示す図である。
【図１６２】２つのピクセルと各ピクセル内の各サンプルの位置を示す図である。
【図１６３】ピクセル内のサンプル間のオフセットを示す図である。
【図１６４】２５６ビット／ピクセルディスプレイマッピングを示す図である。
【図１６５】RPIX、SPIXおよびDPIXオペレーションのための２５６ビット／ピクセルキャッシュライン編成を示す図である。
【図１６６】RDATオペレーションを用いる２５６ビット／ピクセルキャッシュライン編成を示す図である。
【図１６７】8:2* (8:8:8:8:8) :6* (8:8:8:8:32) の２５６ビット／ピクセルフォーマットを示す図である。
【図１６８】8:2* (8:8:8:8:8) :6* (8:8:8:8:32) の２５６ビット／ピクセル表示リフレッシュを示す図である。
【図１６９】SRAMピクセルバッファ読出／書込フォーマットを示す図である。
【図１７０】５１２ビット／ピクセルディスプレイマッピングを示す図である。
【図１７１】RPIX、SPIXおよびDPIXオペレーションのための５１２ビット／ピクセルキャッシュライン編成を示す図である。
【図１７２】RDATオペレーションを用いる５１２ビット／ピクセルキャッシュライン編成を示す図である。
【図１７３】8:2* (8:8:8:8:8) :6* (8:8:8:8:32) の５１２ビット／ピクセルフォーマットを示す図である。
【図１７４】8:2* (8:8:8:8:8) :6* (8:8:8:8:32) の５１２ビット／ピクセル表示リフレッシュを示す図である。
【図１７５】SRAMピクセルバッファ読出／書込フォーマットを示す図である。
【図１７６】8:2* (8:10:10:10) :6* (10:10:10:32) の５１２ビット／ピクセルフォーマットを示す図である。
【図１７７】8:2* (8:2:10:10:10) :6* (2:10:10:10:32) の５１２ビット／ピクセル表示リフレッシュを示す図である。
【図１７８】SRAMピクセルバッファ読出／書込フォーマットを示す図である。
【図１７９】I/Oバスにわたっての高速同時双方向送受信をサポートするデュアルピクセル３DRAMチップの代替のアーキテクチャを示すブロック図である。
【図１８０】I/Oバスにわたっての高速同時双方向送受信をサポートするデュアルピクセル３DRAMチップのための第２の代替のアーキテクチャを示すブロック図である。

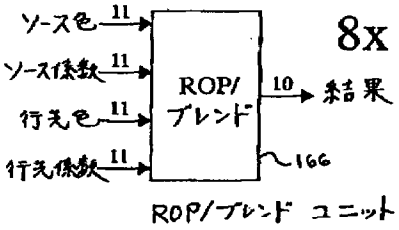
【符号の説明】

１００　グラフィックスサブシステム、１０２　レンダリングコントローラ、１０４　インタフェース、１０６　ビデオ出力回路、１０８　ビデオ出力チャネル、１１０　デュアルピクセル３DRAMチップ、１１２　レンダリングバス、１１４アドレスおよび制御バス、１１６　DRAMアレイ、１１８　SRAMピクセルバッファ、１２０，１２１　ピクセル算術論理演算装置（ALU）、１２２　グローバルバス、１２３　グローバル書込バス、１２４　I/Oバス、１２５　グローバル読出バス、１２６　デマルチプレクサ、１２７　パイプラインレジスタ、１２８　入力データバス、１２９，１３１　入力データバス、１３０　入力データフォーマッタ、１３２　SRAM出力データバス、１３４　出力データフォーマッタ、１３５　出力データバス、１３６　出力データマルチプレクサ、１３７　パイプラインレジスタ、１３８，１３９　データバス、１４０　ピクセルALUからSRAMへのフォーマッタ、１４１　データバス、１４２，１４３　SRAMピクセルバッファデータバス、１４５　書込マスクパイプラインレジスタ、１４６　アドレスおよび制御入力バス、１＊４７　書込マスクバス、１４８　アドレスおよび制御デマルチプレクサ、１５０　アドレスおよび制御バス、１５１　デコーダ、１５２　ピクセルＡＬＵオペレーションチャネル、１５４　グローバルバスオペレーションチャネル、１５６　バンクオペレーションチャネル、１５８　DRAMバンク、１６０センス増幅器、１６６　ROP/ブレンドユニット、１６８　デプスユニット、１７０　ステンシルユニット、１７２　ウィンドウIDユニット、１７４　ROPユニット、１７６　ブレンドユニット、１７８　８ビットブレンドユニット、１８０１０ビットブレンドユニット、１８２　ROPレジスタ、１８４　パターンレジスタ、１８６　ディザ計算装置、１８８，１９０　乗算器、１９２　加算器、１９４　切捨て装置、１９６　クランプ装置、２１０　１６ビットマスクレジスタ、２１６　マスクレジスタ、２１８　基準レジスタ、２２０　ファンクションレジスタ、２２４　キャッシュライン、２２６　ダーティ・タグSRAM、２３０　キャッシュラインのバンクおよびコラムタグ、２３１　バンクパイプラインレジスタ、２３２　コラムパイプラインレジスタ。

【図１】

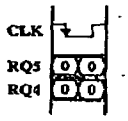



【図７】



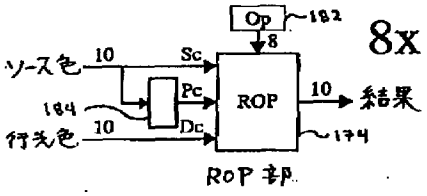

ROP/ブレンド ユニット

【図３０】



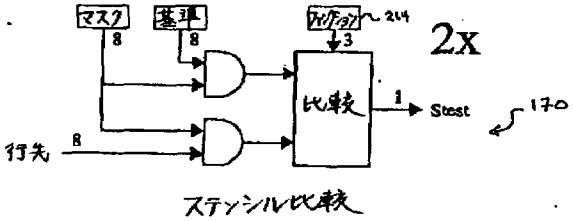

アイドルコマンド

【図８】



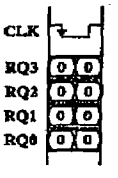

ROP 部

【図１６】



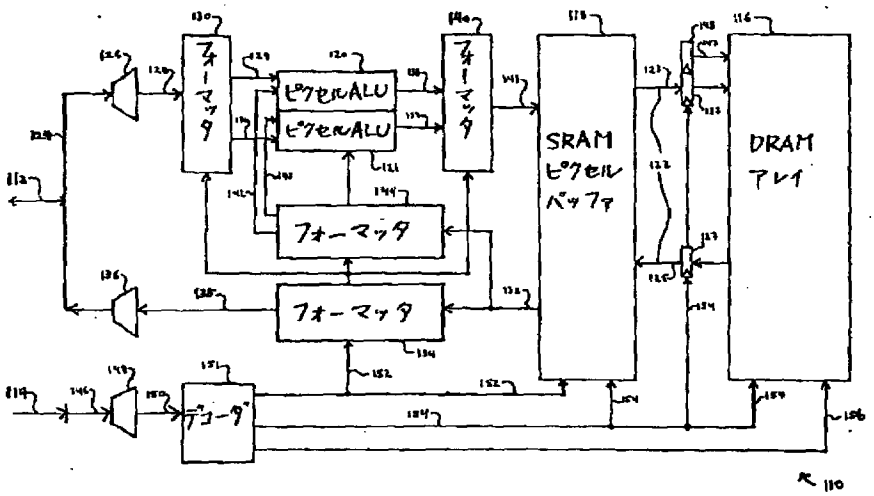

ステンシル比較

【図３８】

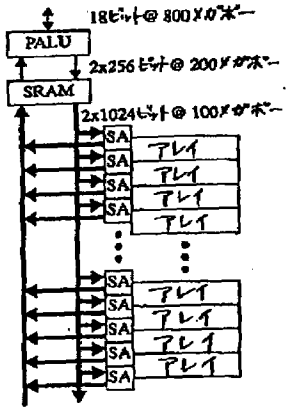



アイドル

【図２】

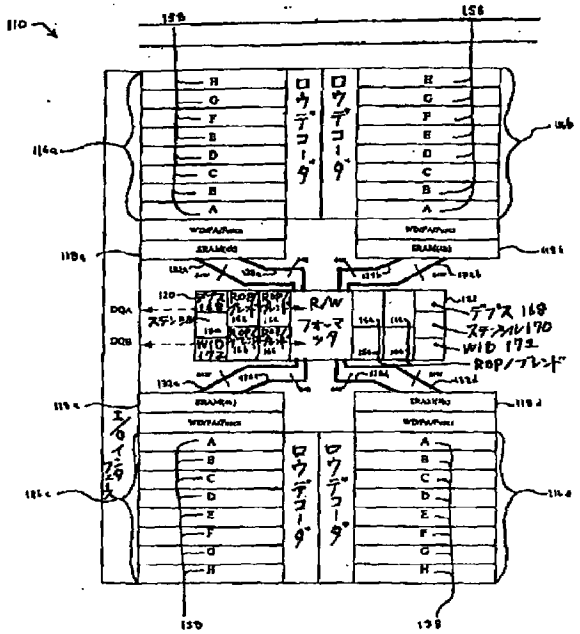



【図２１】

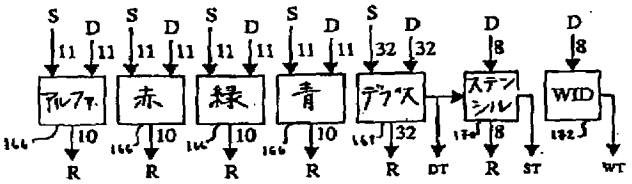



DRAM アレイ メモリ構成

【図３】

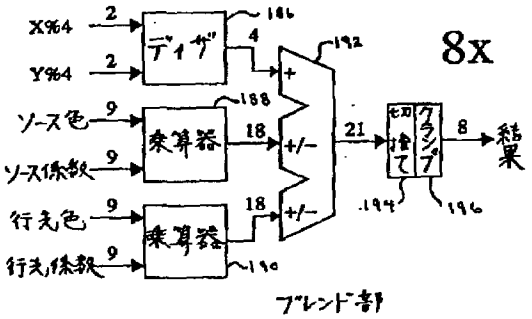



【図６】

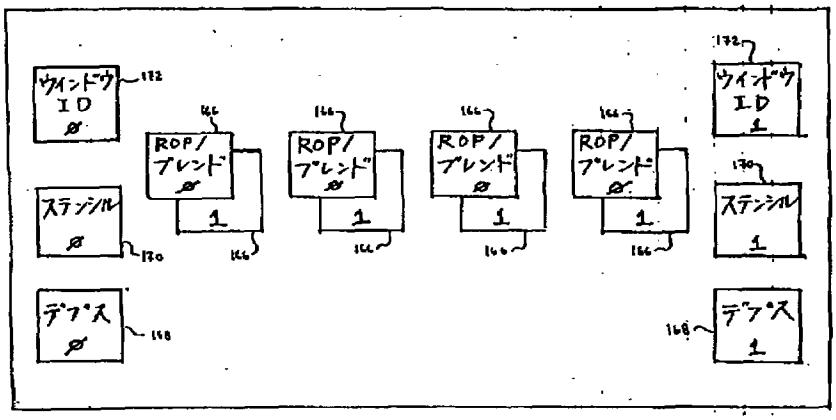



ピクセル ALU の成分

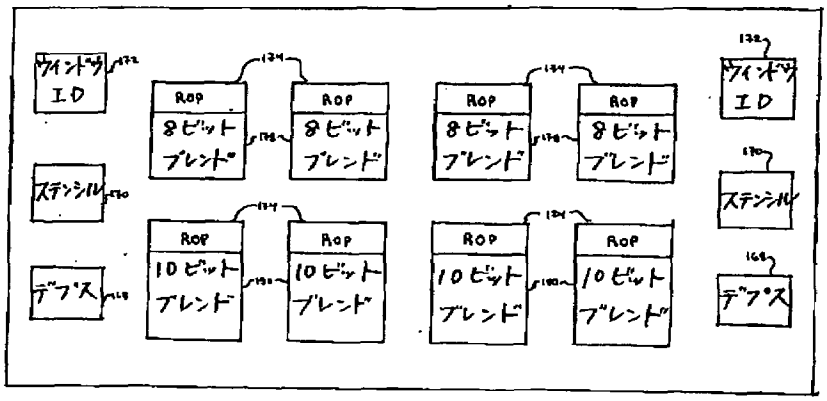
【図９】



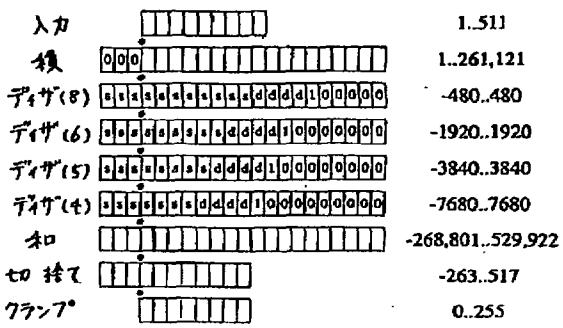
【図４】




【図５】




【図１０】




8ビットブレンドユニットの計算

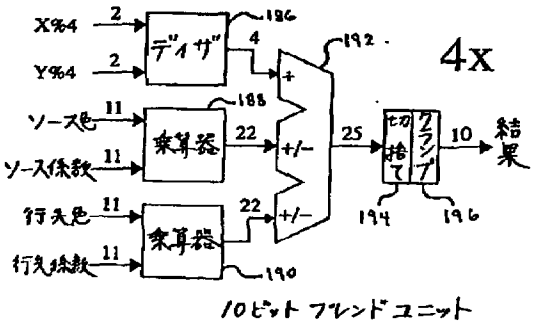
【図１１】

| Y%4 \ X%4 | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | -15/32 | +1/32 | -11/32 | +5/32 |
| 1 | +9/32 | -7/32 | +13/32 | -3/32 |
| 2 | -9/32 | +7/32 | -13/32 | +3/32 |
| 3 | +15/32 | -1/32 | +11/32 | -5/32 |

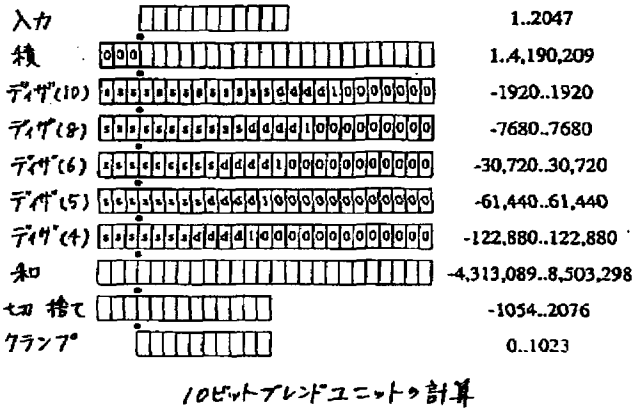
| X%4 | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| | 10001 | 00001 | 10101 | 00101 |
| | 01001 | 11001 | 01101 | 11101 |
| | 10111 | 00111 | 10011 | 00011 |
| | 01111 | 11111 | 01011 | 11011 |

ドット分散型組織的ディザアルゴリズム

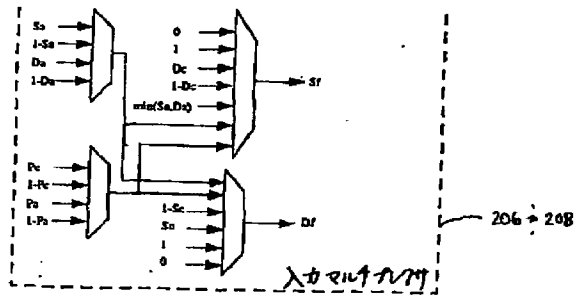
【図１２】



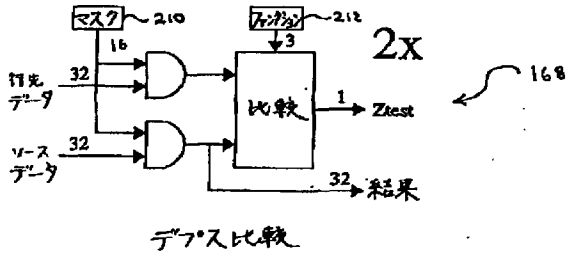

【図１３】



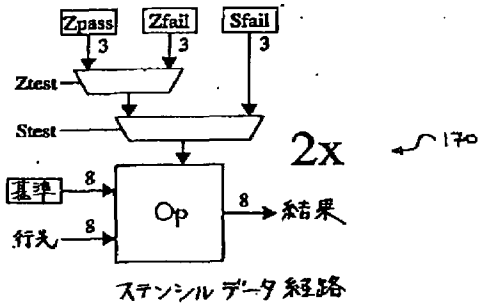

【図１４】



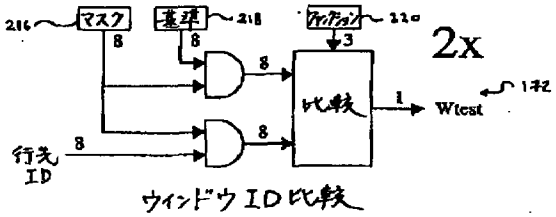

【図１５】



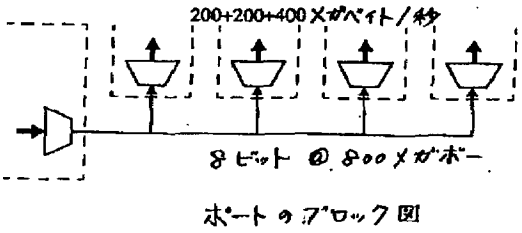

【図１７】



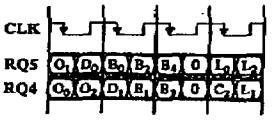

【図１８】



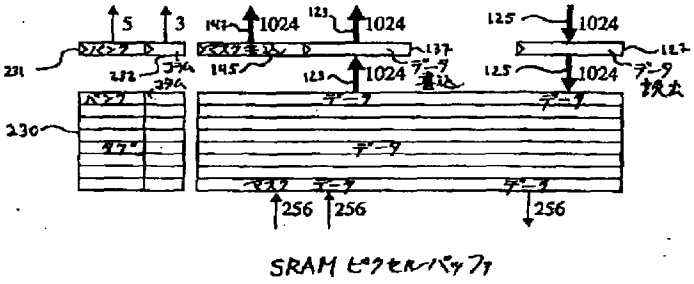

【図２４】



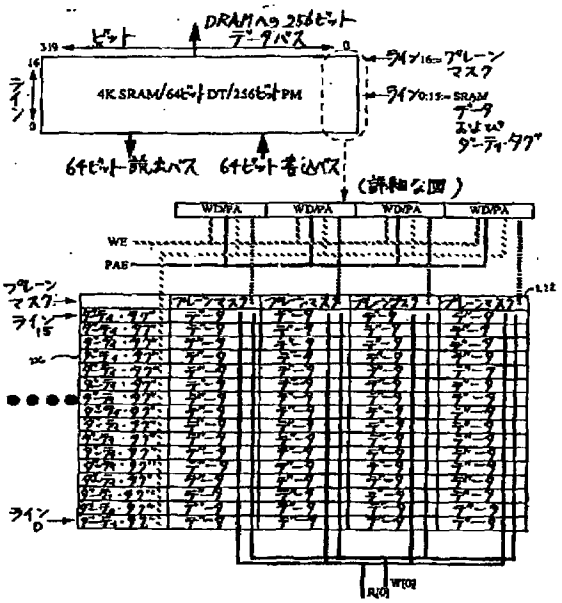

【図３２】



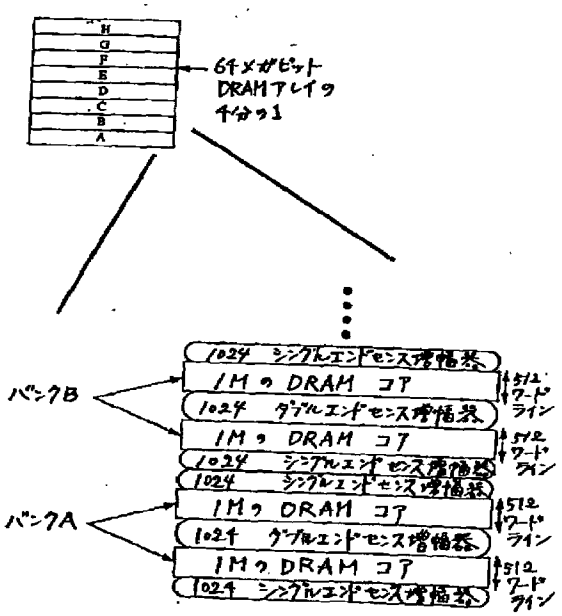
【図１９】



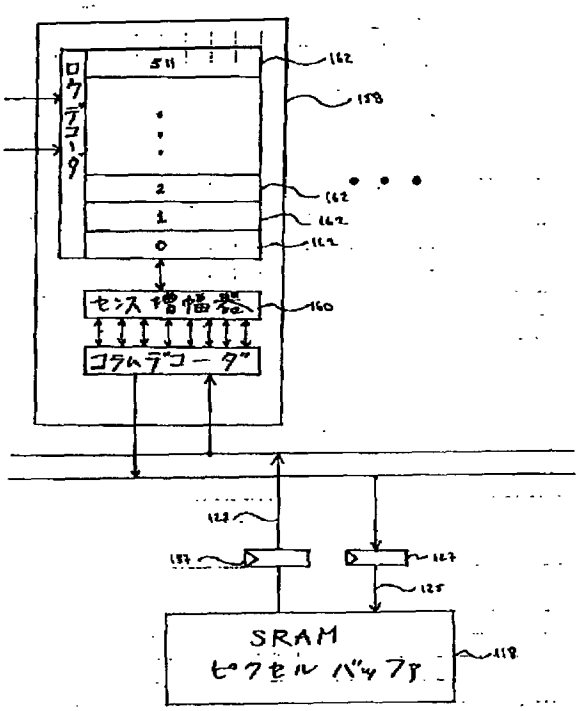

【図２０】



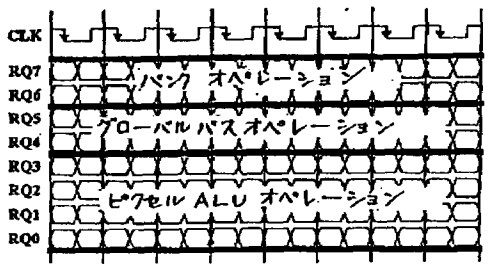
【図２２】




【図２３】

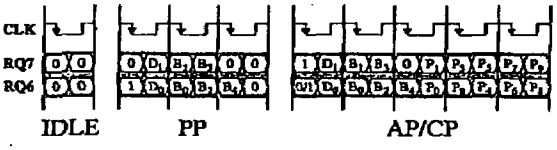



【図２５】



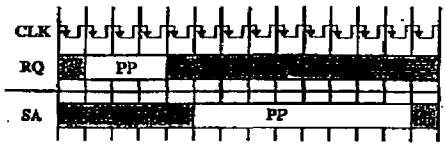

アドレスおよび制御
プロトコル

【図２６】



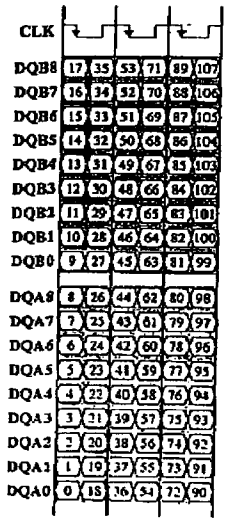

バンク オペレーション
プロトコル

【図２７】



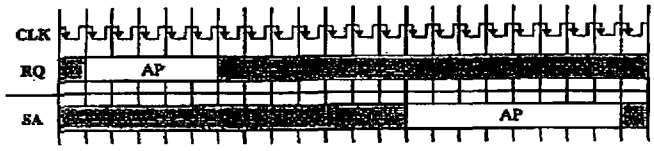

ページ プリチャージ

【図４２】



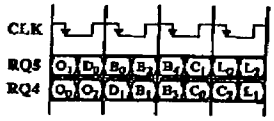

データ転送

【図２８】




ページ アクセス

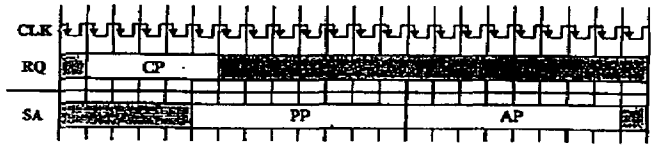
【図３１】




RLコマンド、
WLコマンド、
MLコマンド、
および
CLコマンド

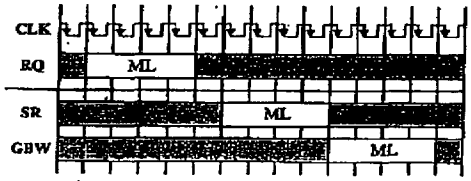
【図２９】

CP(ページ変更)は APオペレーションが続く
PPオペレーションを組み合わせる



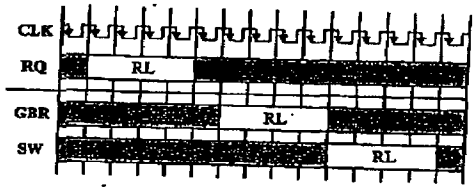

ページ変更

【図３５】



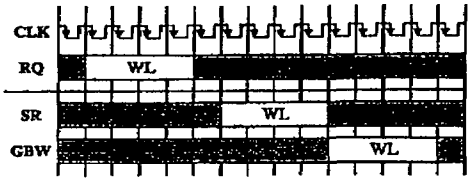

ML オペレーション

【図３３】



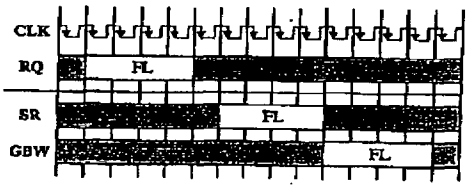

RL オペレーション

【図３４】



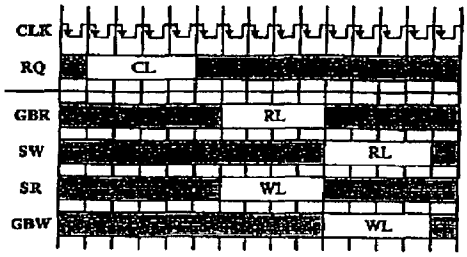

WL オペレーション

【図３６】



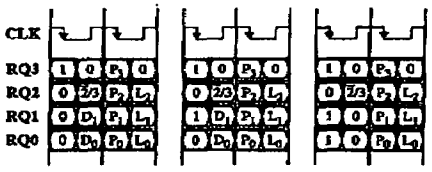

FL オペレーション

【図３７】



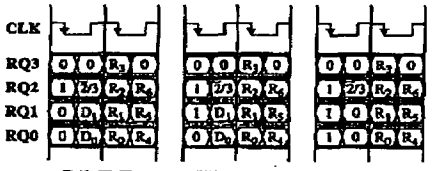

CL オペレーション

【図３９】



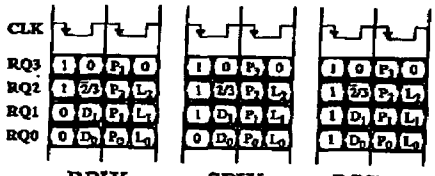

RDAT, WDAT, BDAT

【図４０】



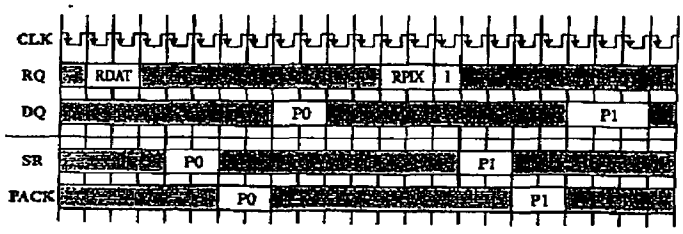

RREG, WREG, BREG

【図４１】



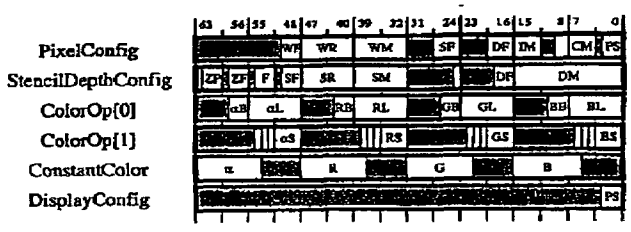

RPIX, SPIX, DPIX

【図４３】



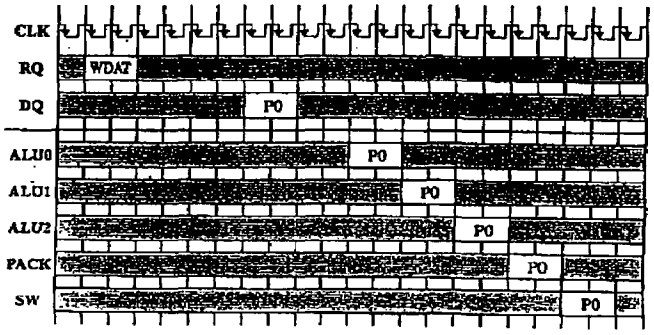

2サイクル RDAT オペレーション
続いて３サイクル RPIX オペレーション

【図５４】



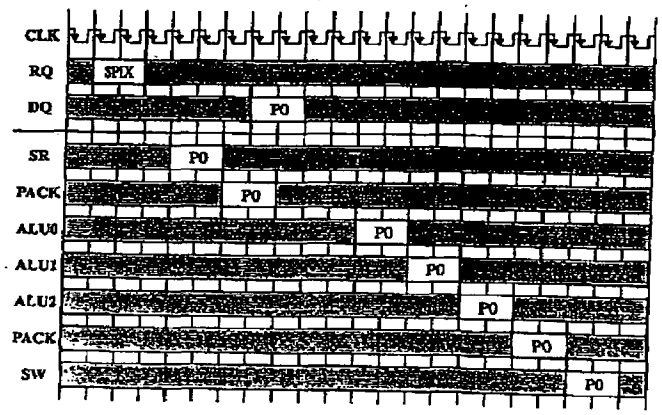

レジスタ データ フォーマット

【図４４】



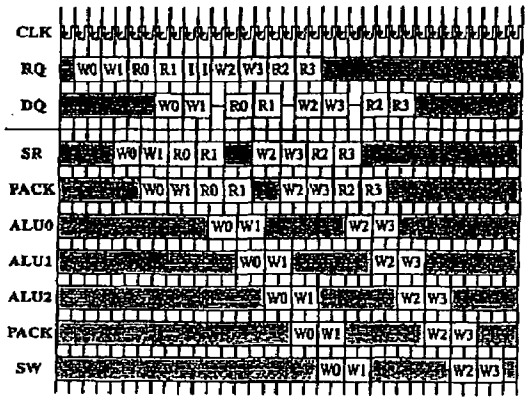

WDAT, BDAT, WREG, BREG
書込およびブロードキャスト
オペレーション

【図４５】



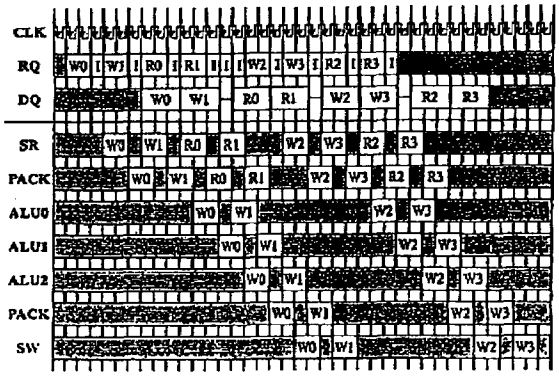

SPIX, DPIX
ピクセル オペレーション

【図４７】



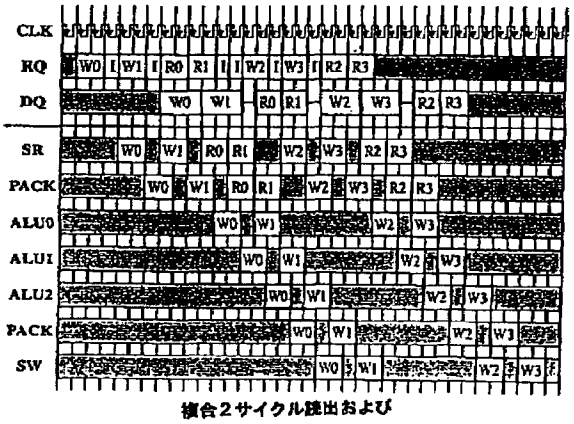

複合２サイクル読出および
２サイクル書込オペレーション

【図５０】



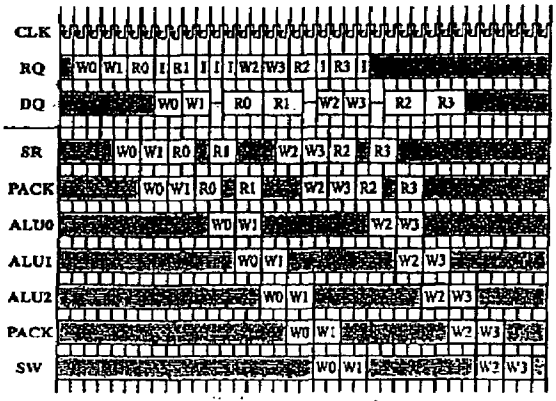

複合３サイクル読出および
３サイクル書込オペレーション

【図４８】



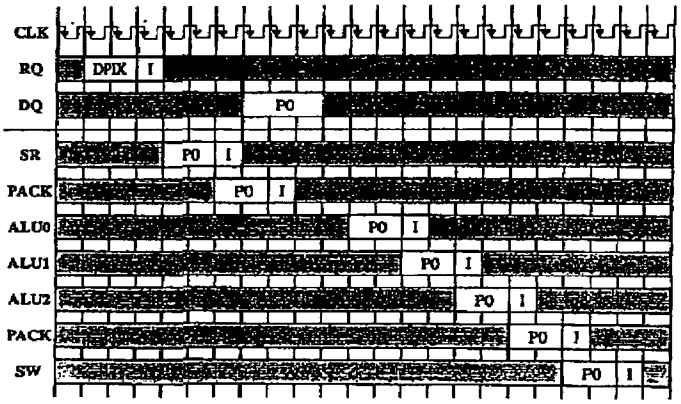

複合２サイクル読出および
３サイクル書込オペレーション

【図４９】



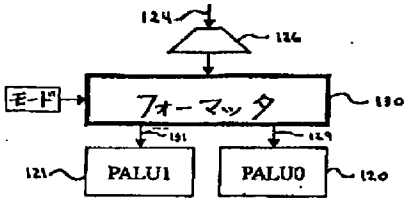

複合３サイクル読出および
２サイクル書込オペレーション

【図４６】



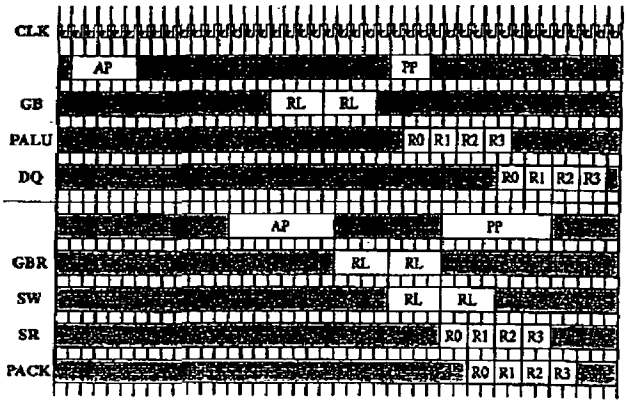

3サイクルのDPIX転送

【図６３】



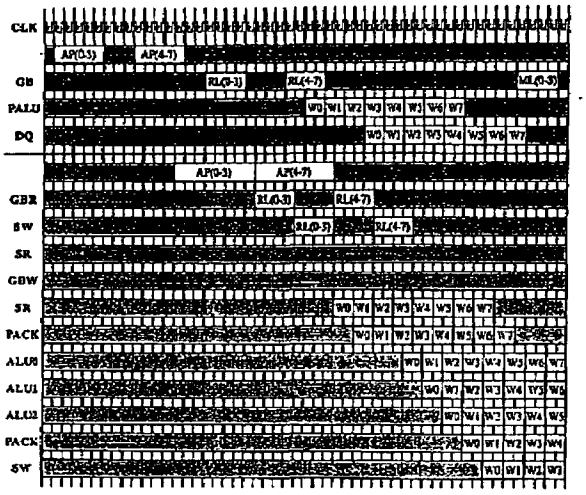

DQからピクセルALUへのルーティング

【図５１】



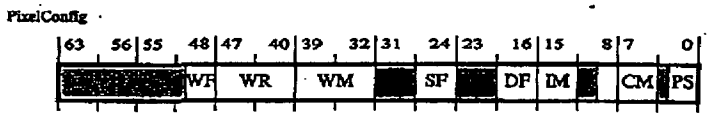

4つの2サイクル読出オペレーション

【図５２】



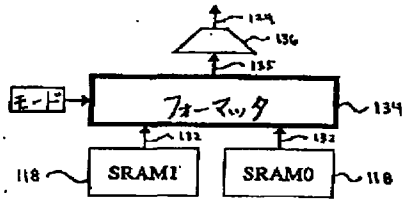

8つの2サイクルDPIXオペレーション

【図５５】



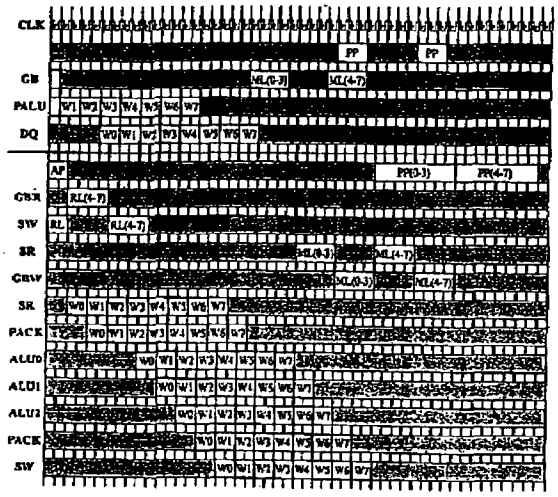

PixelConfig レジスタのデータフォーマット

【図６８】



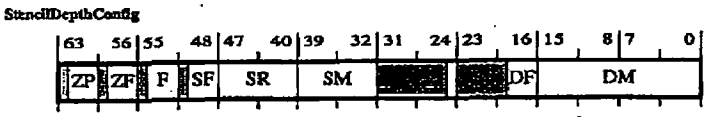

SRAMからDQへのルーティング

【図５３】



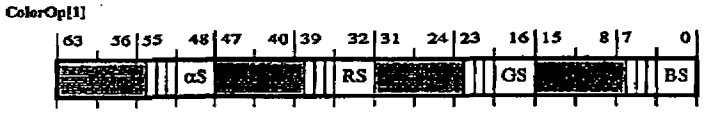

8つ２サイクルDPIXオペレーション
（続き）

【図５６】



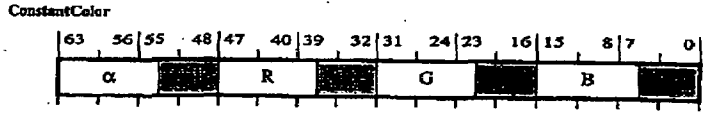

StencilDepthConfig レジスタのデータフォーマット

【図５８】



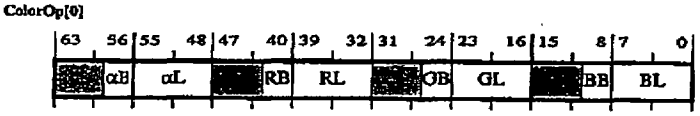

ColorOp[1] レジスタのデータフォーマット

【図５９】



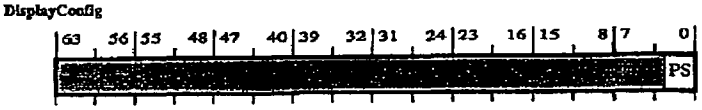

ConstantColor レジスタのデータフォーマット

【図５７】



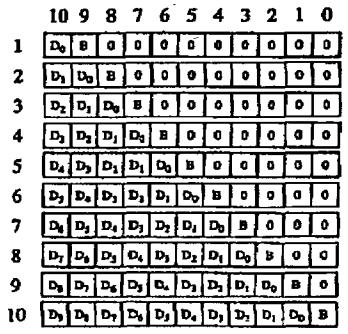

ColorOp[0] レジスタのデータフォーマット

【図６０】



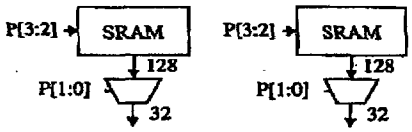

DisplayConfig レジスタのデータフォーマット

【図６７】



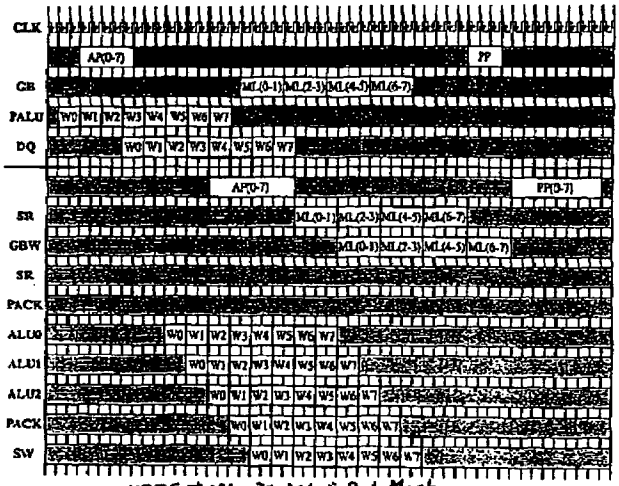

ROP/Blend ユニットのための
色成分 フォーマッティング

【図７０】



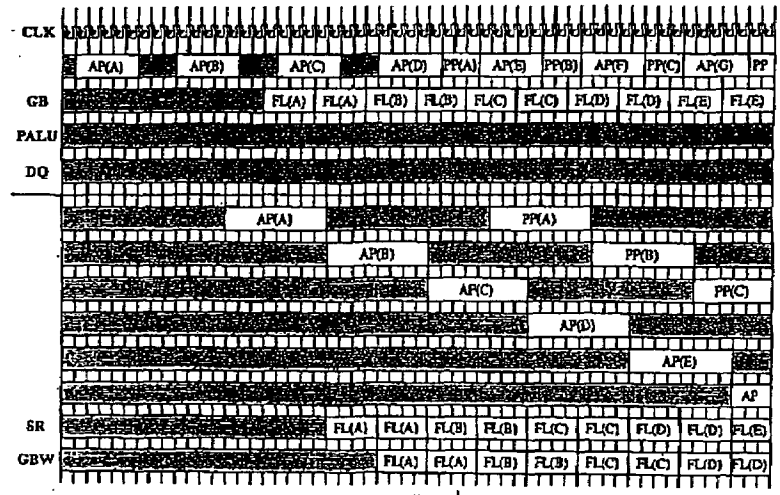

SRAMからDQへのルーティング

【図６１】



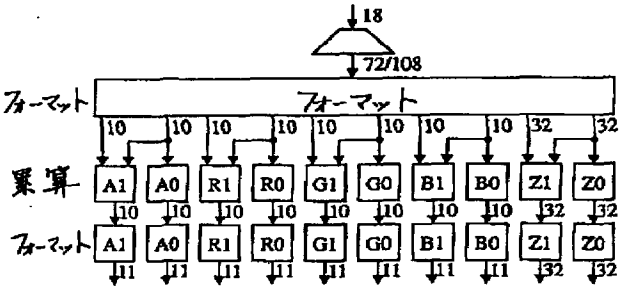

WREGオペレーションからByteMask
および MLオペレーションが高速フィルを行う

【図６２】



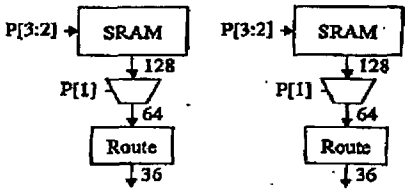

FLオペレーションが非常に高速なフィルを行う

【図６４】



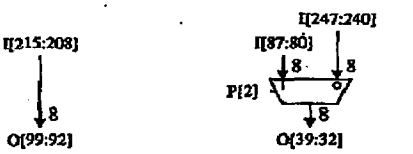

入力 ルーティング層

【図７２】



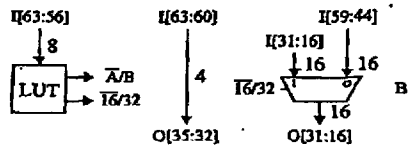

SRAMからDQへのルーティング

【図７７】



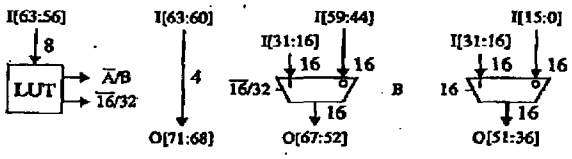

SRAMからDQへのルーティング

【図７３】



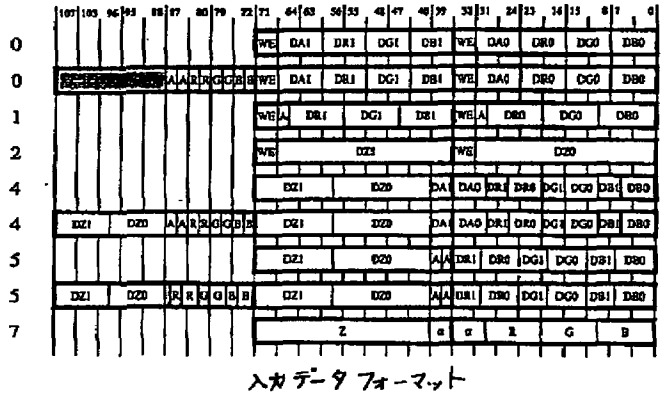

SRAMからDQへのルーティング

【図７４】

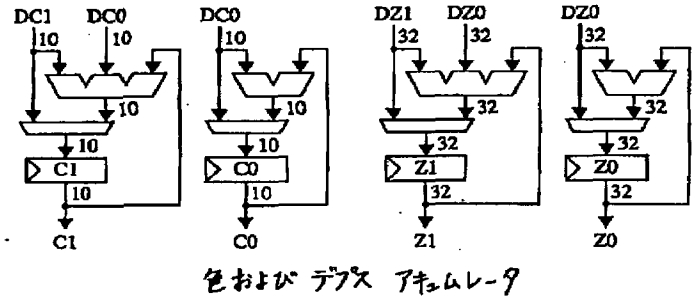



SRAMからDQへのルーティング

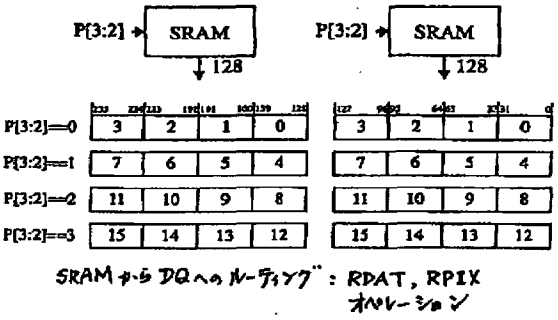
【図６５】



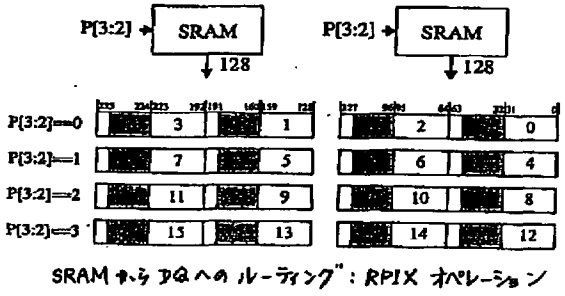

【図６６】



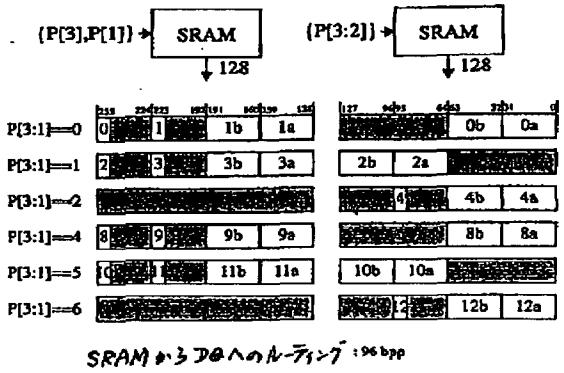

【図６９】



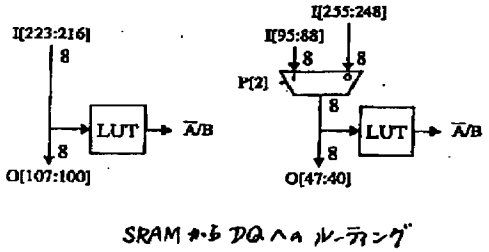

【図７１】



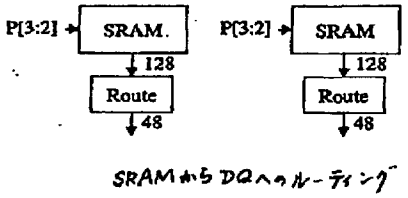

【図７５】



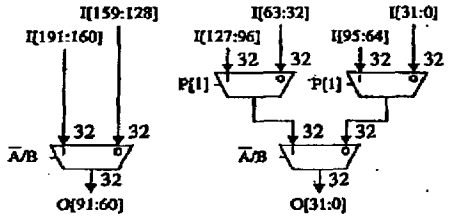

【図７６】




【図８０】

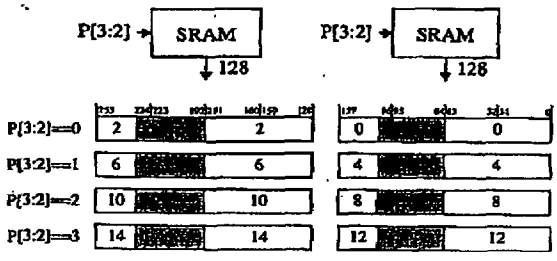


【図７８】



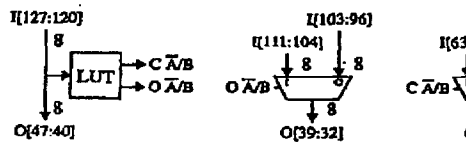

SRAMからDQへのルーティング

【図７９】



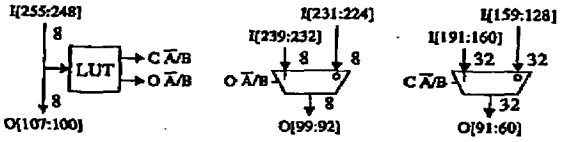

SRAMからDQへのルーティング

【図８１】



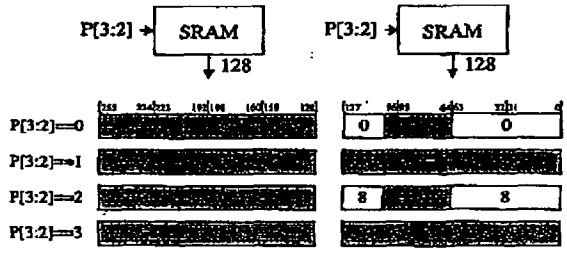

SRAMからDQへのルーティング

【図８２】



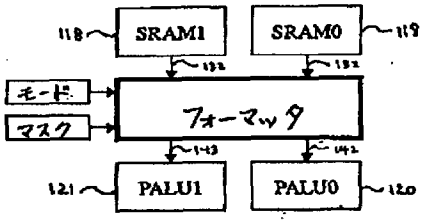

SRAMからDQへのルーティング

【図８３】




SRAMからDQへのルーティング

【図８４】




SRAMからピクセルALUへのルーティング

【図９８】

【図８５】

| | | | |
|---|---|---|---|
| 3c | 2c | 1c | 0c |
| 7c | 6c | 5c | 4c |
| 11c | 10c | 9c | 8c |
| 15c | 14c | 13c | 12c |

| | | | |
|---|---|---|---|
| 3c | 2c | 1c | 0c |
| 7c | 6c | 5c | 4c |
| 11c | 10c | 9c | 8c |
| 15c | 14c | 13c | 12c |

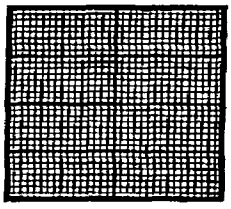
SRAM編成 8、16、32 bpp

【図８６】

| | | | |
|---|---|---|---|
| 3de | 3ab | 1de | 1ab |
| 7de | 7ab | 5de | 5ab |
| 11de | 11ab | 9de | 9ab |
| 15de | 15ab | 13de | 13ab |

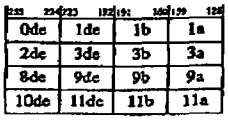
| | | | |
|---|---|---|---|
| 2dc | 2ab | 0dc | 0ab |
| 6dc | 6ab | 4dc | 4ab |
| 10dc | 10ab | 8dc | 8ab |
| 14dc | 14ab | 12dc | 12ab |

SRAM編成 64 bpp



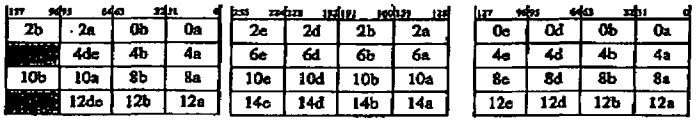
8 bpp ディスプレイマッピング

【図８７】




SRAM 編成 96 bpp

【図８８】

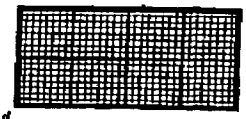



SRAM 編成 128 bpp

【図１０１】

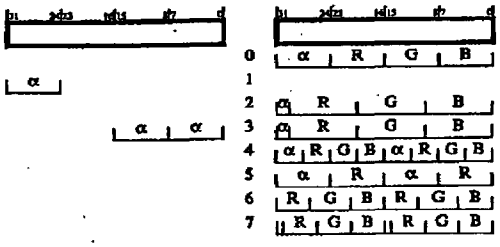
8 bpp ピクセルフォーマット

【図１０３】



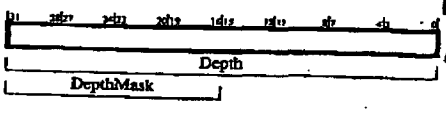
16 bpp ディスプレイマッピング

【図８９】




Unpack 関数

【図９０】



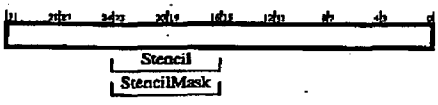

UnpackDepth

【図１００】

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| RDAT 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RDAT 1 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RDAT 2 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RDAT 3 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| RDAT 4 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| RDAT 5 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 |
| RDAT 6 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 |
| RDAT 7 | 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 |
| RDAT 8 | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 |
| RDAT 9 | 79 | 78 | 77 | 76 | 75 | 74 | 73 | 72 |
| RDAT 10 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | 80 |
| RDAT 11 | 95 | 94 | 93 | 92 | 91 | 90 | 89 | 88 |
| RDAT 12 | 103 | 102 | 101 | 100 | 99 | 98 | 97 | 96 |
| RDAT 13 | 111 | 110 | 109 | 108 | 107 | 106 | 105 | 104 |
| RDAT 14 | 119 | 118 | 117 | 116 | 115 | 114 | 113 | 112 |
| RDAT 15 | 127 | 126 | 125 | 124 | 123 | 122 | 121 | 120 |

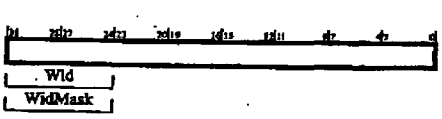
RDATを用いる 8 bpp キャッシュライン編成

【図９１】



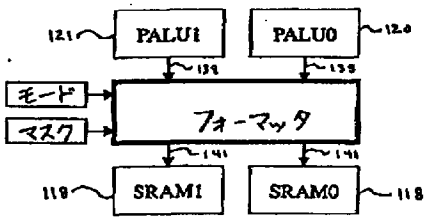

UnpackStencil

【図９２】



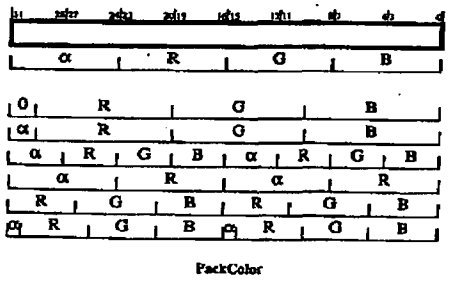

UnpackWid

【図９３】



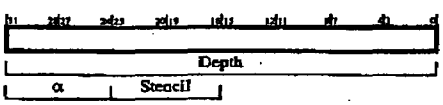

ピクセル ALU から SRAM へのルーティング

【図９４】



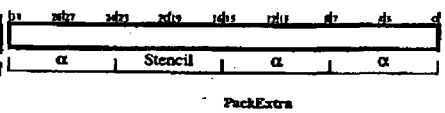

PackColor

【図９５】




PackDepth

【図９６】

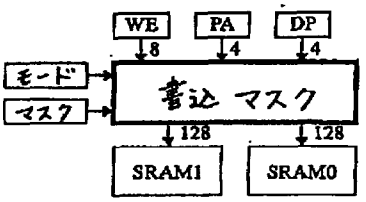



PackExtra

【図１０６】

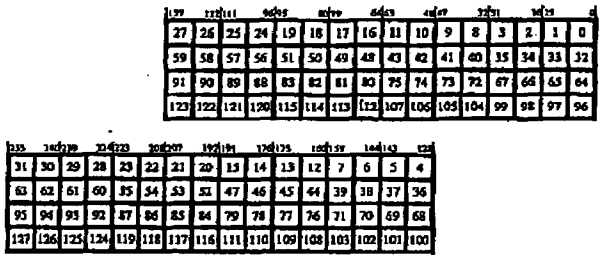
16 bpp ピクセルフォーマット 4:4:4:4

【図９７】



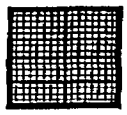

ピクセルALUから SRAMへの マスク生成

【図９９】



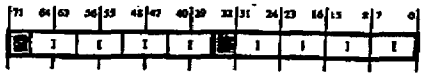
RPIX、SPIX、DPIX のための 8 bpp キャッシュライン編成

【図１１２】



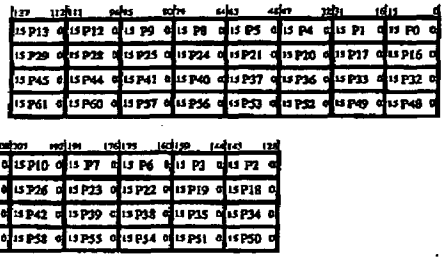
32 bpp ディスプレイマッピング

【図１０２】



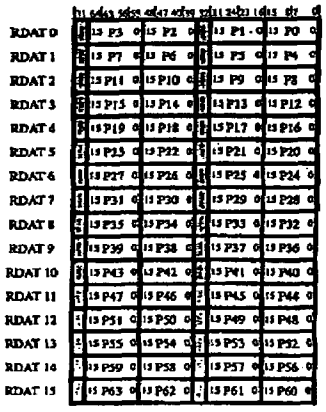
8 bpp 表示リフレッシュ

【図１０４】



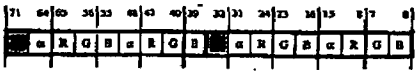
RPIX、SPIX、DPIX のための キャッシュライン編成

【図１０５】




RDATを用いる
16 bpp キャッシュライン編成

【図１０７】



16 bpp 表示リフレッシュ 4:4:4:4

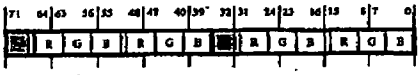
【図１０８】

16 bpp ピクセルフォーマット 5:6:5

【図１１０】

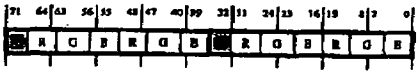
16 bpp ピクセルフォーマット 1:5:5:5

【図１１１】



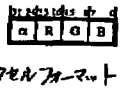
16 bpp 表示リフレッシュ 1:5:5:5

【図１０９】



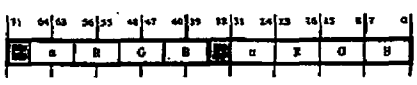
16 bpp 表示リフレッシュ 5:6:5

【図１１５】



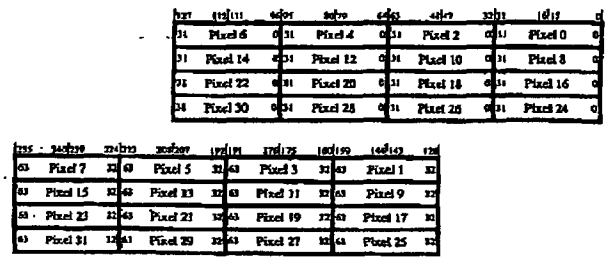
32 bpp ピクセルフォーマット 8:8:8:8

【図１１６】



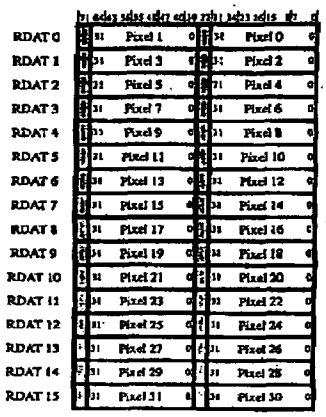
32 bpp 表示リフレッシュ 8:8:8:8

【図１１３】



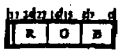

RPIX、SPIX、DPIXのための、32 bpp キャッシュライン編成

【図１１４】




RDATを用いる
32 bpp キャッシュライン編成

【図１１７】

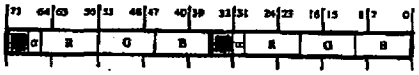



32bppピクセルフォーマット 10:10:10

【図１２２】

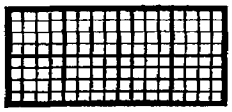

64 bppピクセルフォーマット 4:8:8:8:8

【図１１８】



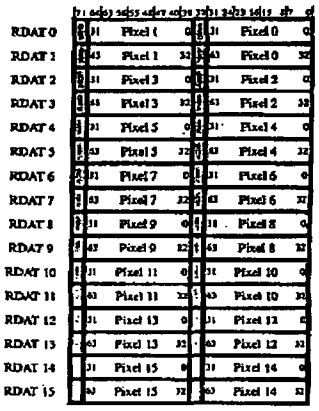

RGB 2:10:10:10 表示リフレッシュ

【図１１９】



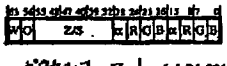
64bpp ディスプレイマッピング

【図１２１】



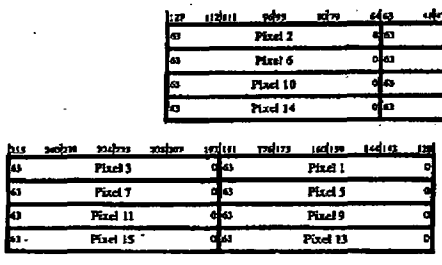

RDATを用いる
64 bpp キャッシュライン編成

【図１２８】



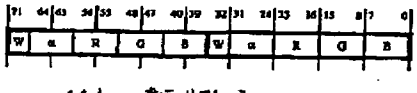

64 bpp ピクセルフォーマット 4:14:24:12*(4:4:4:4)

【図１２０】



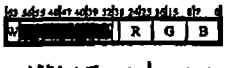

RPIX、SPIX、DPIXのための、64 bpp キャッシュライン編成

【図１２３】



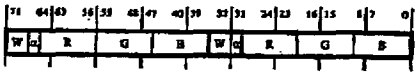

64 bpp 表示リフレッシュ 4:8:8:8:8

【図１２４】



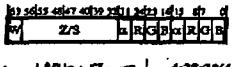

64 bppピクセルフォーマット 2:10:10:10

【図１２５】



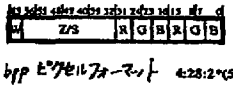

64 bpp 表示リフレッシュ 4:10:10:10

【図１２６】



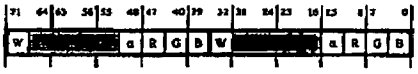

64 bppピクセルフォーマット 4:28:2*(4:4:4:4)

【図１３０】



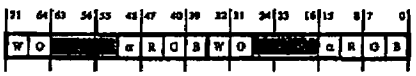

64 bpp ピクセルフォーマット 4:28:2*(5:6:5)

【図１２７】



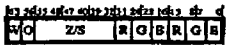
64 bpp 表示リフレッシュ 4:28:2*(4:4:4:4)

【図１２９】



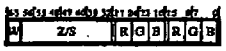
64 bpp 表示リフレッシュ 4:4:24:2*(4:4:4:4)

【図１３２】



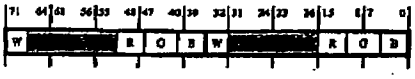
64 bpp ピクセルフォーマット 4:4:24:2*(5:6:5)

【図１３４】



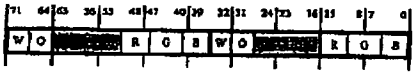
64 bpp ピクセルフォーマット 4:28:2*(1:5:5:5)

【図１３１】



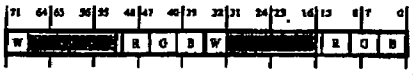
64 bpp 表示リフレッシュ 4:28:2*(5:6:5)

【図１３３】



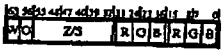
64 bpp 表示リフレッシュ 4:4:24:2*(5:6:5)

【図１３５】



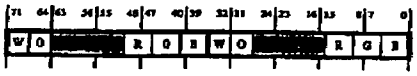
64 bpp 表示リフレッシュ 4:28:2*(1:5:5:5)

【図１３６】



64 bpp ピクセルフォーマット 4:4:24:2*(1:5:5:5)

【図１３７】

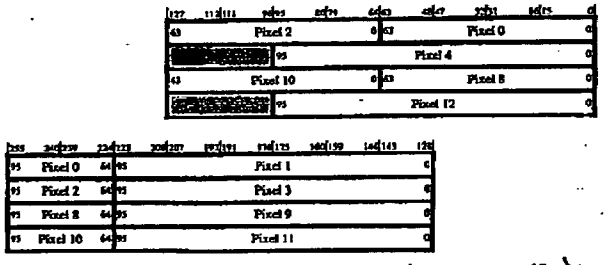


64 bpp 表示リフレッシュ 4:4:24:2*(1:5:5:5)

【図１３８】

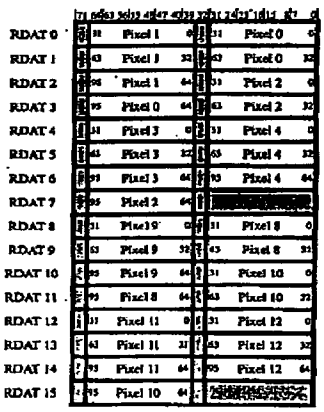
96 bpp ディスプレイマッピング

【図１３９】



RPIX、SPIX、DPIXのための、96 bpp キャッシュライン編成

【図１４０】



RDATを用いる
96 bpp キャッシュライン編成

【図１４１】

96 bpp ピクセルフォーマット 4:28:2*(8:8:8:8)

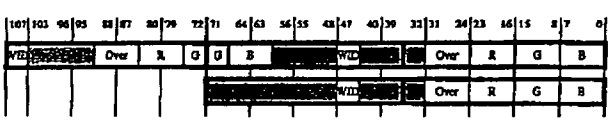
【図１４３】

96 bpp ピクセルフォーマット 4:28:2*(8:8:8:8)

【図１４５】

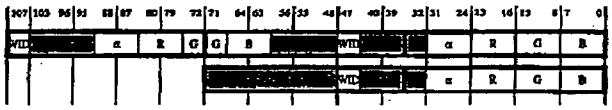
96 bpp ピクセルフォーマット 4:4:24:2*(8:8:8:8)

【図１４２】



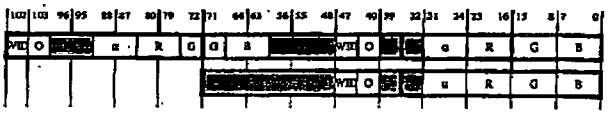
96 bpp 表示リフレッシュ 4:28:2*(8:8:8:8)

【図１４４】



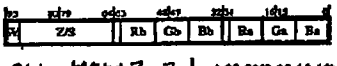
96 bpp 表示リフレッシュ 4:28:2*(8:8:8:8)

【図１４６】



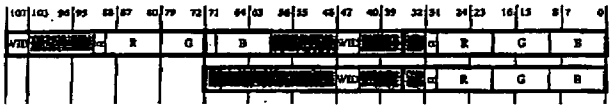
96 bpp 表示リフレッシュ 4:4:24:2*(8:8:8:8)

【図１４７】



96 bpp ピクセルフォーマット 4:28:2*(2:10:10:10)

【図１４８】

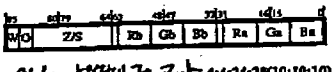


96 bpp 表示リフレッシュ 4:28:2*(2:10:10:10)

【図１５３】

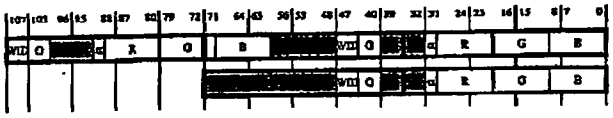
128 bpp ディスプレイマッピング

【図１４９】



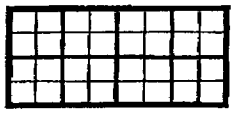
96 bpp ピクセルフォーマット 4:4:24:2*(10:10:10)

【図１５０】



96 bpp 表示リフレッシュ 4:4:24:2*(10:10:10)

【図１６４】



256 bpp ディスプレイマッピング

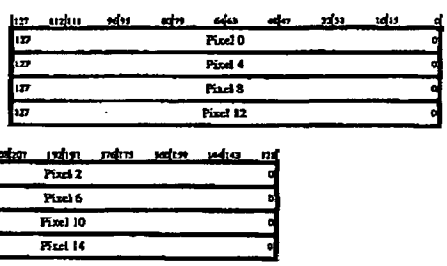
【図１５１】

96 bpp ピクセルフォーマット 4:4:24:4*(4:4:4:4)

【図１５２】

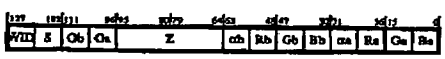
96 bpp 表示リフレッシュ 4:4:24:4*(4:4:4:4)

【図１５４】



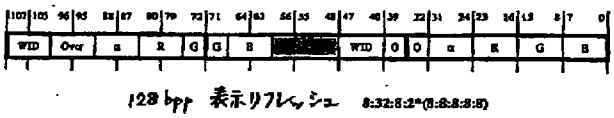

RPIX、SPIX、DPIXのための、128 bpp キャッシュライン編成

【図１５６】



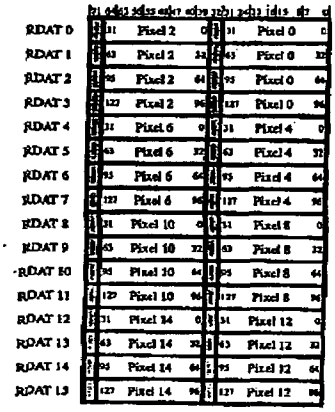
128 bpp ピクセルフォーマット 8:32:8:2*(8:8:8:8)

【図１５７】



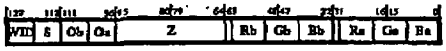
128 bpp 表示リフレッシュ 8:32:8:2*(8:8:8:8)

【図１５５】




RDATを用いる
128 bpp キャッシュライン構成

【図１５８】




128 bpp ピクセルフォーマット 8:32:8:2*(8:2:10:10:10)

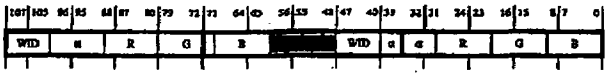
【図１６０】


128 bpp ピクセルフォーマット 8:8:32:2*(10:10:10:10)

【図１７０】

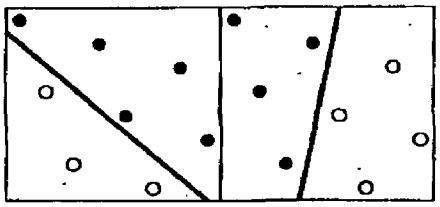
512 bpp ディスプレイマッピング

【図１６１】

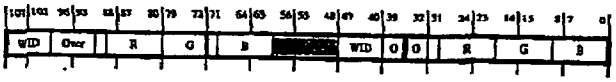



128 bpp 表示リフレッシュ 8:8:32:2*(10:10:10:10)

【図１６２】

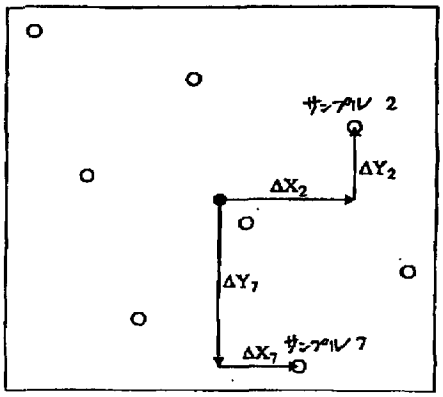


【図１５９】

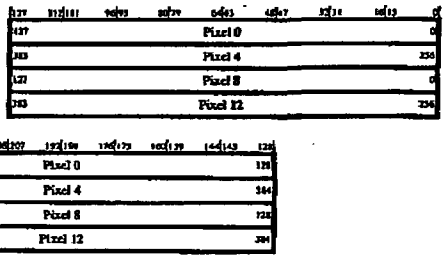



128 bpp 表示リフレッシュ 8:32:8:2*(8:10:10:10)

【図１６３】

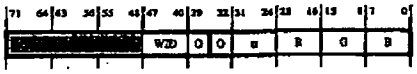



【図１６５】



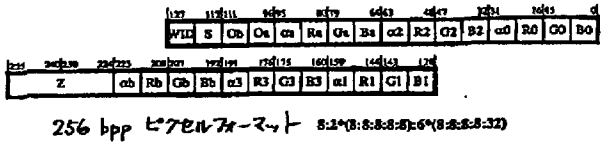

RPIX、SPIX、DPIX のための、256 bpp キャッシュライン構成

【図１６８】



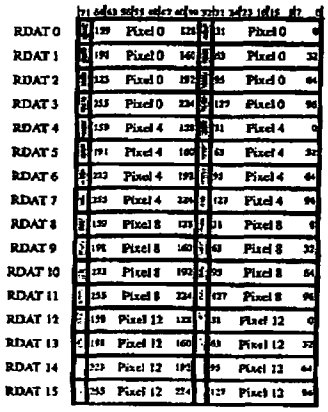

256 bpp 表示リフレッシュ 8:2*(8:8:8:8:8):6*(8:8:8:8:32)

【図１６７】



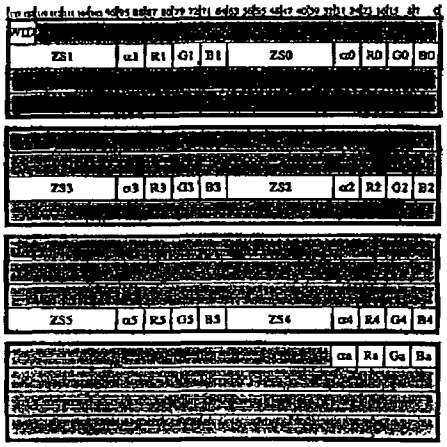

256 bpp ピクセルフォーマット 8:2*(8:8:8:8:8):6*(8:8:8:8:32)

【図１６６】



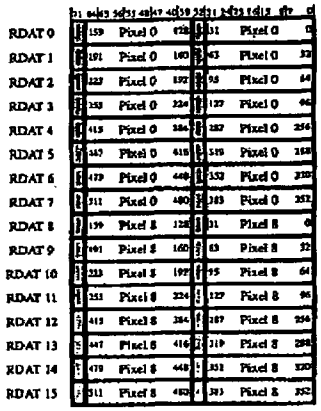

RDATを 用いる
256 bpp キャッシュライン編成

【図１６９】



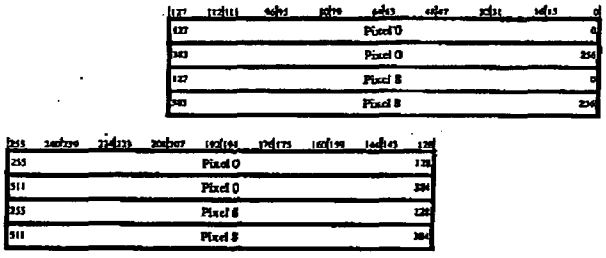

SRAM 読出/書込フォーマット

【図１７２】



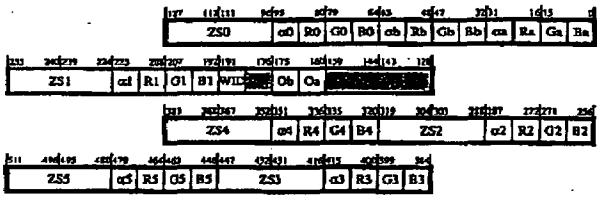

RDATを用いる
512 bpp キャッシュライン編成

【図１７１】



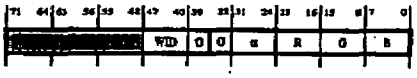

RPIX、SPIX、DPIXのための、512 bpp キャッシュライン編成

【図１７３】



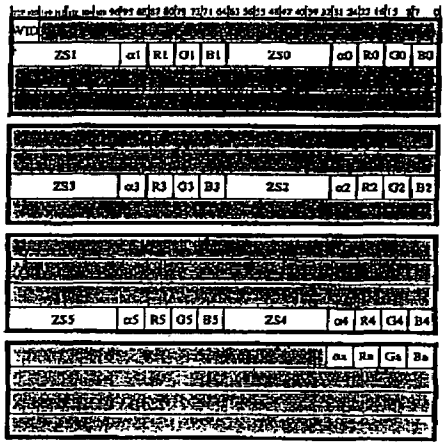

512 bpp ピクセルフォーマット 8:2*(8:8:8:8:8):6*(8:8:8:8:32)

【図１７４】



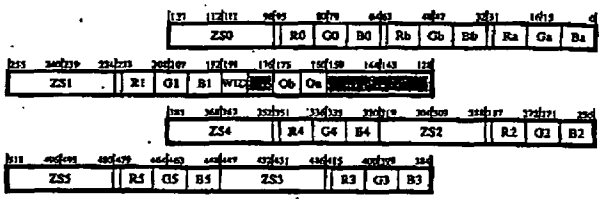

512 bpp 表示リフレッシュ 8:2*(8:8:8:8:8):6*(8:8:8:8:32)

【図１７５】



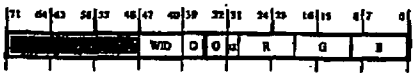

SRAM 読出/書込フォーマット

【図１７６】



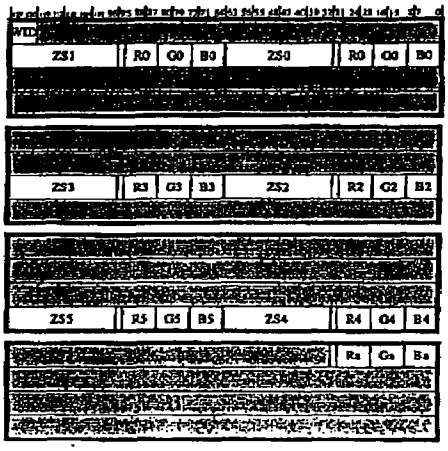

5/2 bpp ピクセルフォーマット 8:2*(8:10:10:10):6*(10:10:10:32)

【図１７７】



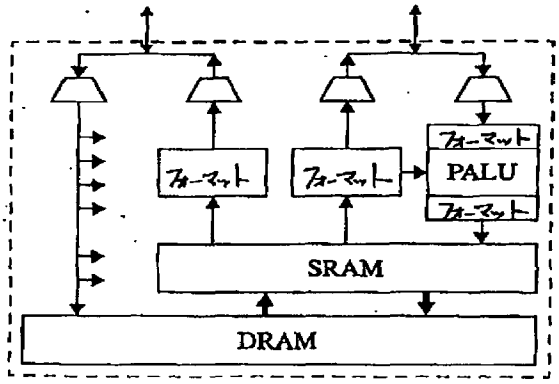

5/2 bpp 表示リフレッシュ 8:2*(8:2:10:10:10):6*(2:10:10:10:32)

【図１７８】

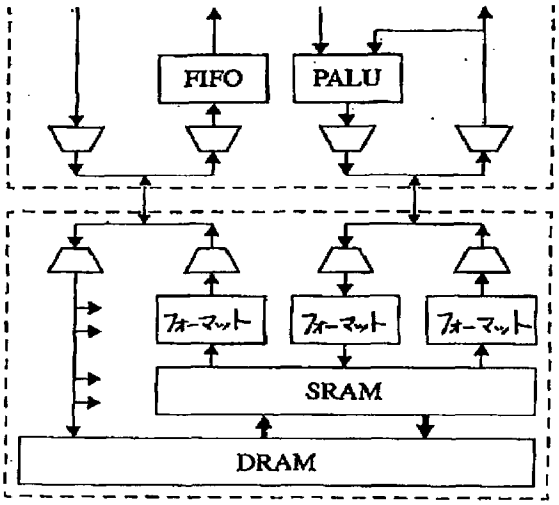



SRAM 読出/書込フォーマット

【図１７９】




【図１８０】



フロントページの続き

(31)優先権主張番号　０９／２６４２６１
(32)優先日　平成11年３月８日(1999．３．８)
(33)優先権主張国　米国（ＵＳ）
＊(31)優先権主張番号　０９／２６４２８１
(32)優先日　平成11年３月８日(1999．３．８)
＊(33)優先権主張国　米国（ＵＳ）

(54)【発明の名称】　記憶装置、データフォーマッタ、データにアクセスする方法、データの領域をクリアする方法、データを圧縮する方法、データをフォーマット化する方法、グラフィックスシステムおよびグラフィックスシステムを動作させる方法